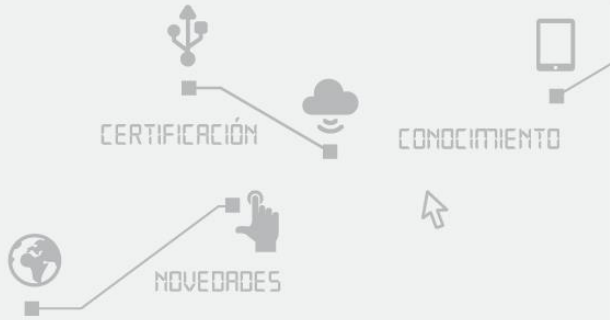
PUE DAY
ABRIL 2016



10:15 - 11:20

Workshop Oracle

"Curso oficial Java Foundations: implementación curricular"

Jordi Ariño, Responsable Desarrollo Software, PUE

ORACLE ACADEMY

pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

Cisco Networking Academy · Linux Professional Institute · cloudera · vmware IT ACADEMY · pue mobile CERTIFICATION · ORACLE ACADEMY · Microsoft Imagine Academy · Scrum Manager

PUE DAY

WORKSHOP ORACLE

"Curso oficial Java Foundations: implementación curricular"

Jordi Ariño, Responsable Desarrollo Software, PUE

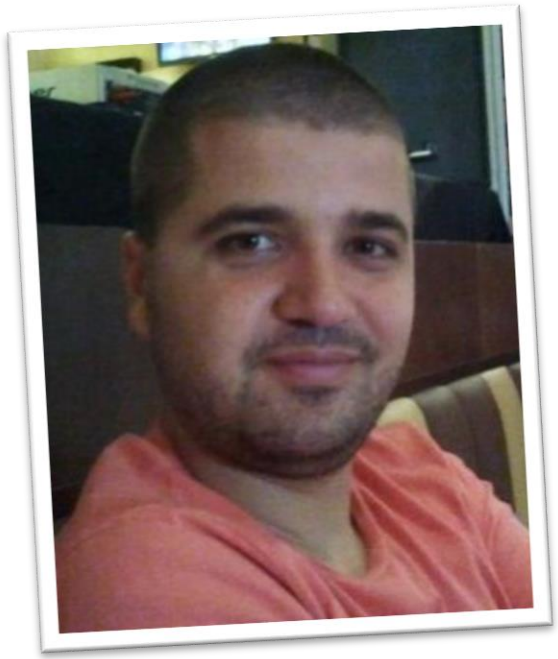
MADRID - 13/04/2016

Curso oficial Java Foundations: implementación curricular

Jordi Ariño Santos – *Agile Developer at PUE*
jordi.arino@pue.es

pue

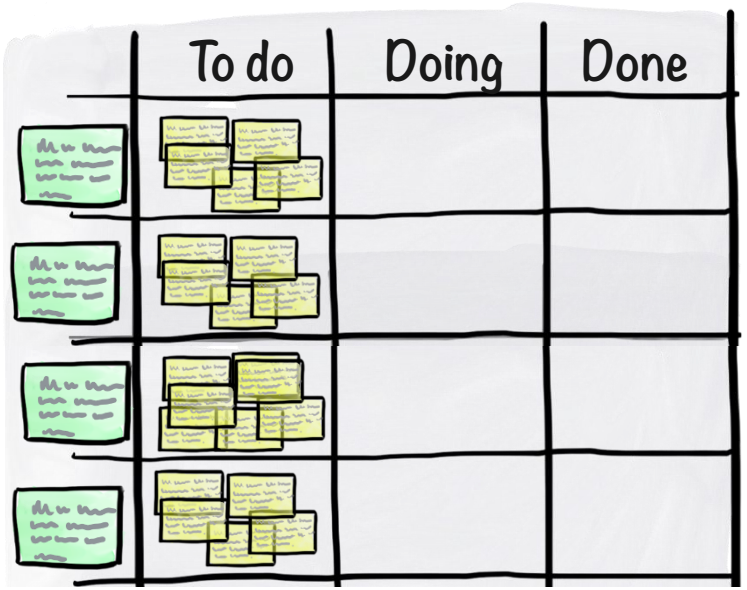
Acerca de...



Jordi Ariño

Agile Developer at PUE
Certified ScrumMaster (CSM)

jordi.arino@pue.es
@jordiAS2K

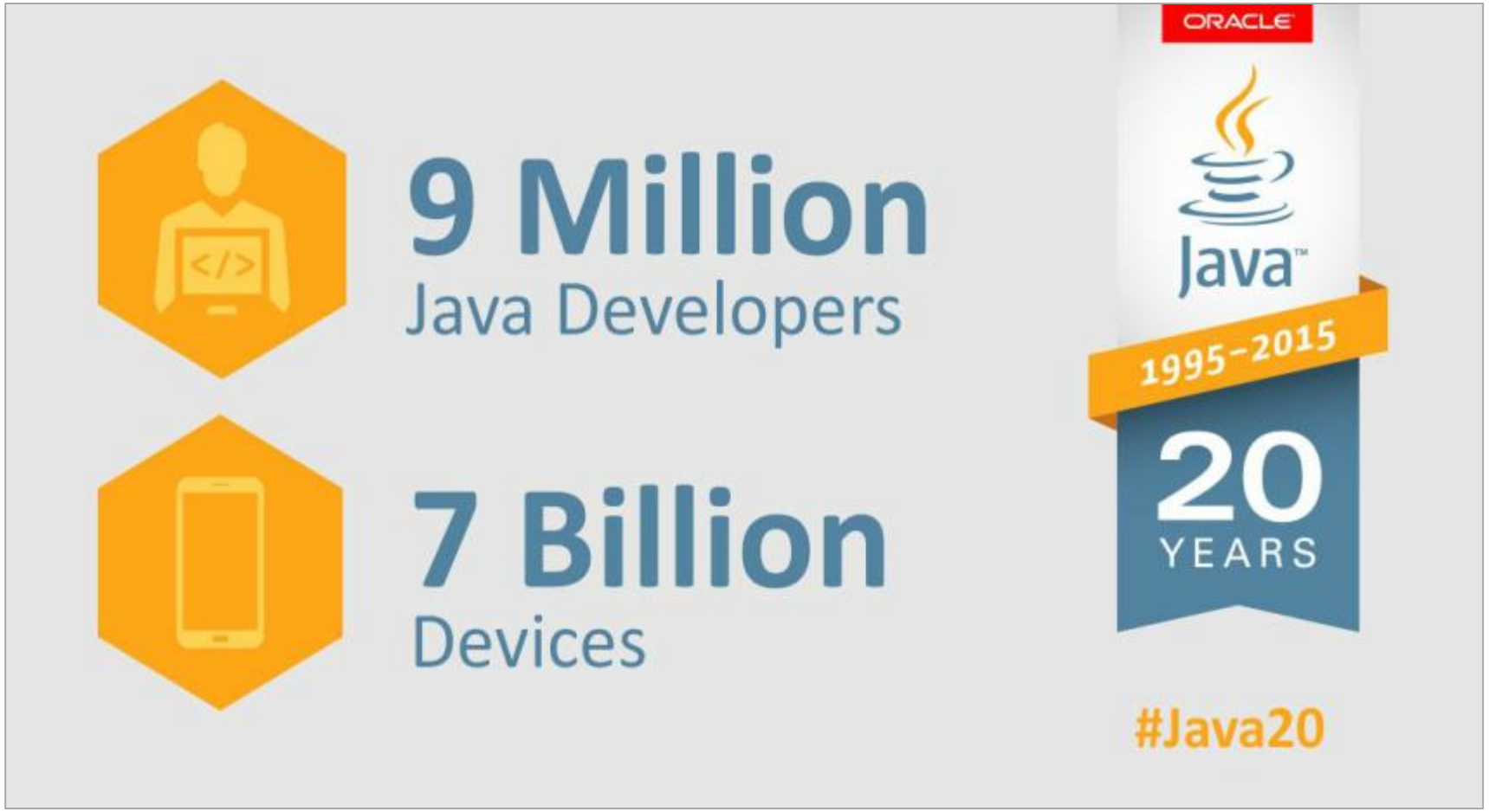


Kanban Board
(Visual Management)

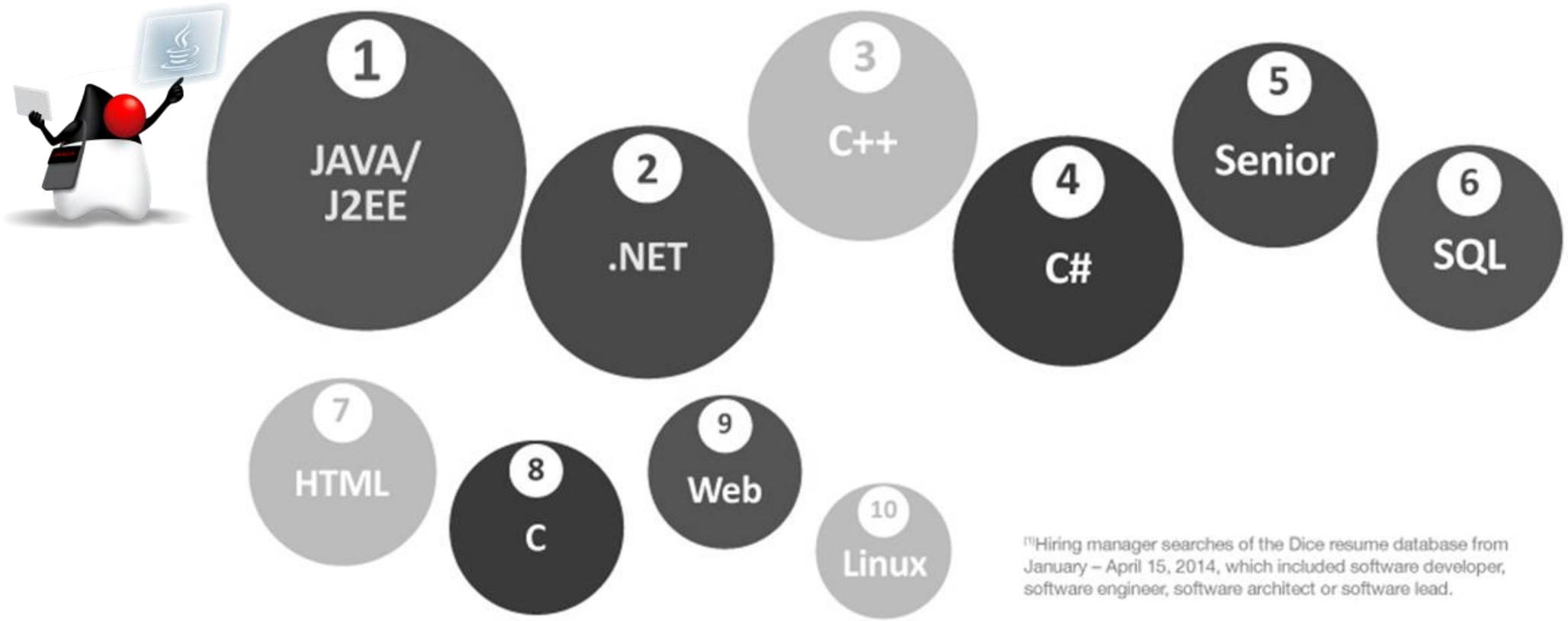
Agenda

1.- Introducción
2.- Plan de formación y certificación en Java
3.- Curso Java Foundations
4.- Estructura del curso Java Foundations
5.- Demo: Curso Java Foundations
6.- Oracle iLearning: Informes y diplomas

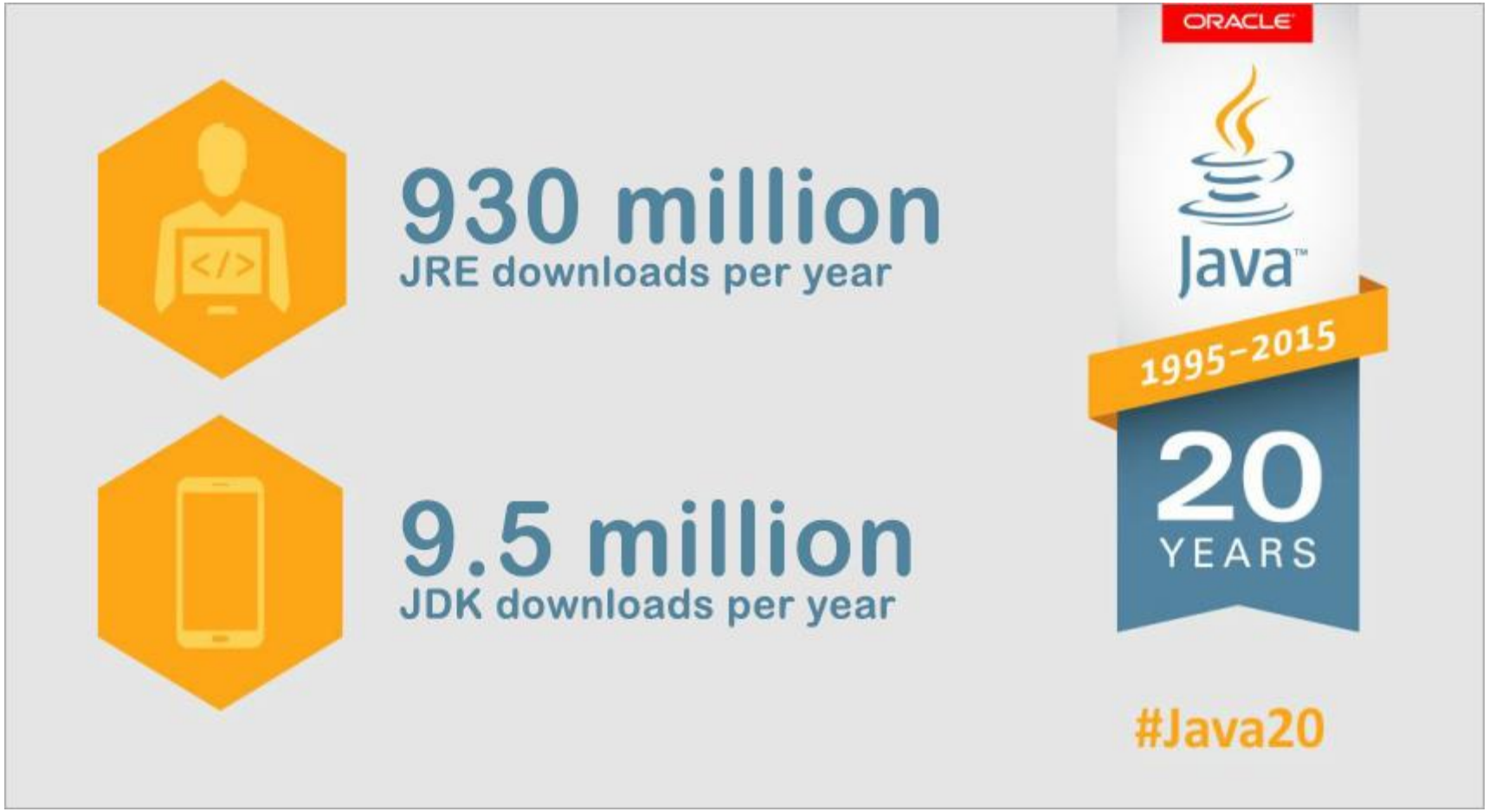
Introducción



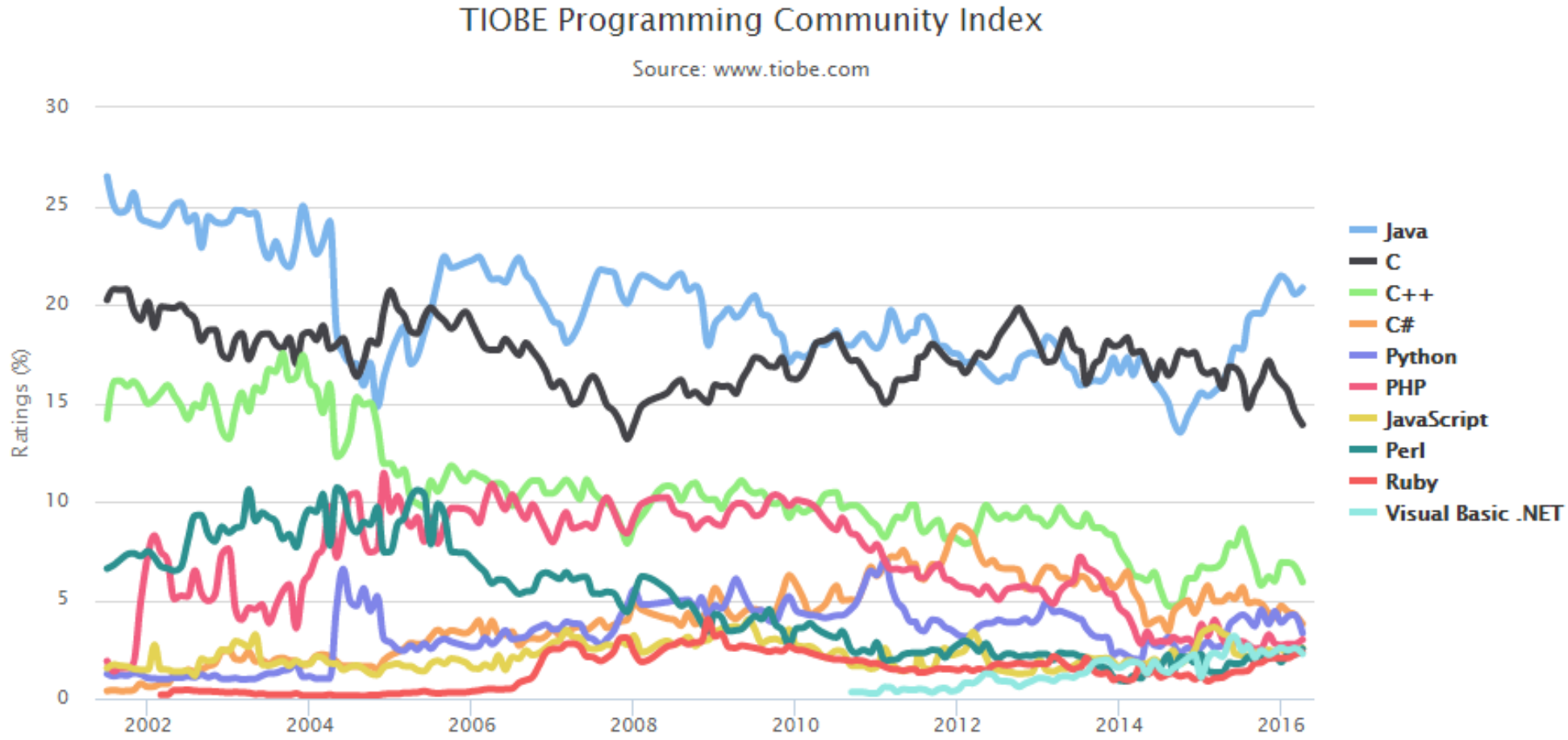
Introducción

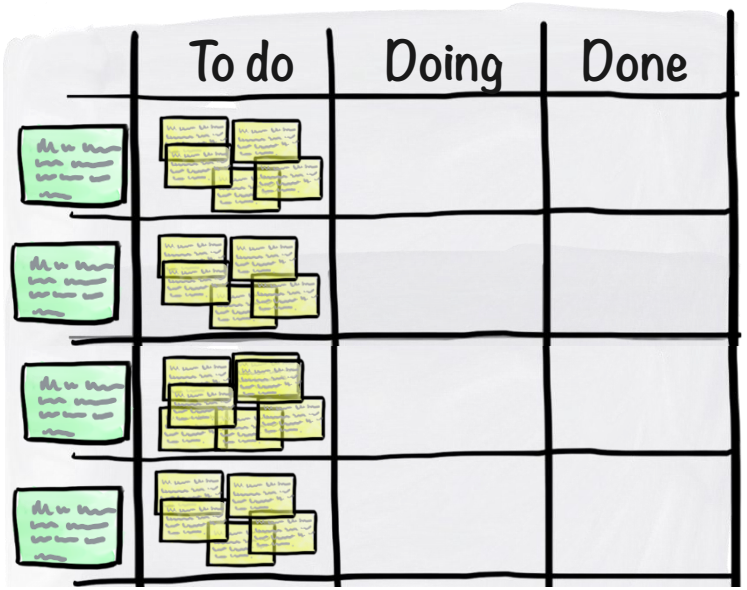


Introducción



Introducción





Kanban Board
(Visual Management)

Agenda

1.- Introducción
2.- Plan de formación y certificación en Java
3.- Curso Java Foundations
4.- Estructura del curso Java Foundations
5.- Demo: Curso Java Foundations
6.- Oracle iLearning: Informes y diplomas

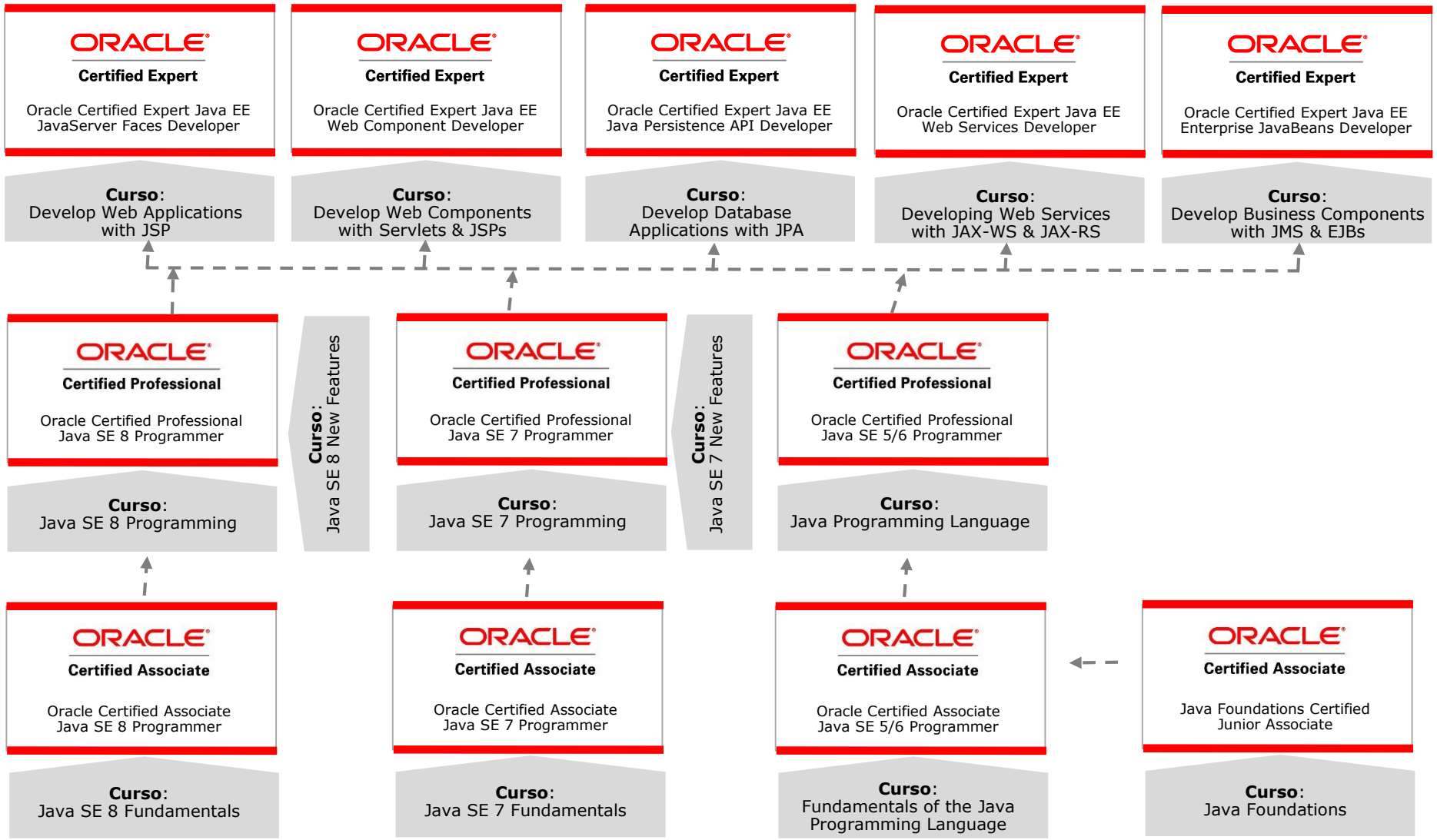
Plan de formación y certificación en Java

La credencial ***Oracle Certified Associate (OCA)*** valida las **habilidades** y **conocimientos fundamentales** que forman una **base sólida de conocimiento** en los productos de Oracle. Convertirse en Oracle Certified Associate es el primer paso que se debe tomar para convertirse en un profesional certificado de Oracle.

La credencial ***Oracle Certified Professional (OCP)*** es el punto de referencia de la **cualificación profesional** y **experiencia técnica** en productos de Oracle. Valida la capacidad para administrar, desarrollar o implementar una amplia gama de productos de Oracle, incluyendo bases de datos, desarrollo de aplicaciones en Java y más. En las empresas a menudo se utiliza la credencial OCP para evaluar las habilidades y experiencia de los empleados y/o posibles candidatos.

La credencial ***Oracle Certified Expert (OCE)*** valida **experiencia** y **habilidades en tecnologías específicas**, arquitecturas o dominios que no están cubiertos específicamente dentro del path de certificaciones estándar o tradicional.

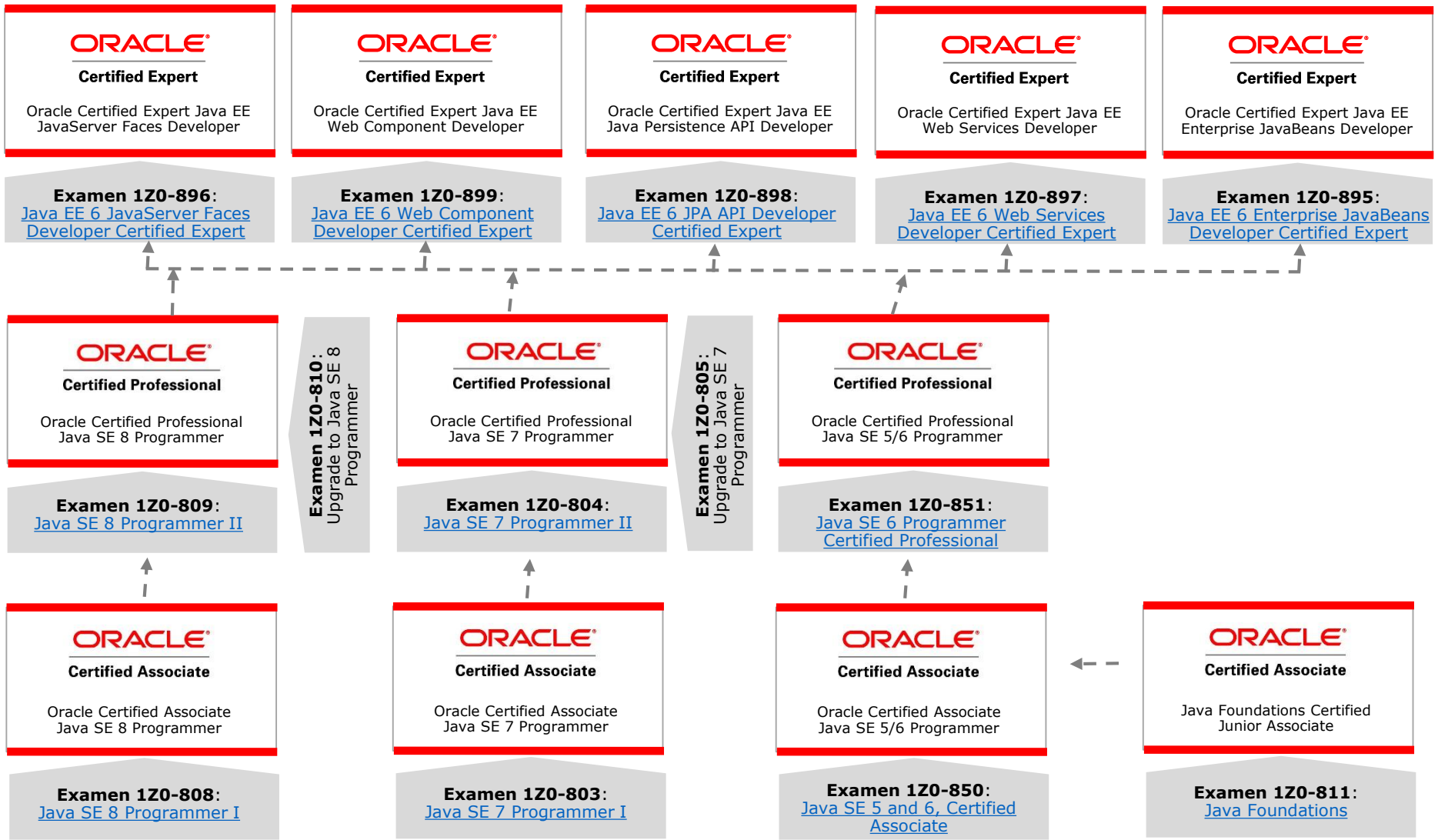
Plan de formación y certificación en Java



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Plan de formación y certificación en Java



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Plan de formación y certificación en Java

ORACLE
Certified Professional

Oracle Certified Professional
Java SE Programming

Curso:
Java Programming

ORACLE
Certified Associate

Oracle Certified Associate
Java SE Programming

Curso:
Java Fundamentals

ORACLE
Certified Associate

Java Foundations Certified
Junior Associate

Curso:
Java Foundations

Creating Java Programs
with Greenfot

Getting started with Java
using Alice

Plan de formación y certificación en Java

Workshop Descriptions

Getting Started with Java Using Alice

This workshop is designed for students with little or no programming experience and teaches basic Java programming concepts through developing 3-D Animations in Alice 3.1.
Students will have fun creating animated stories and games using objects and characters from a rich gallery of 3-D models.

Duration 8 Hours

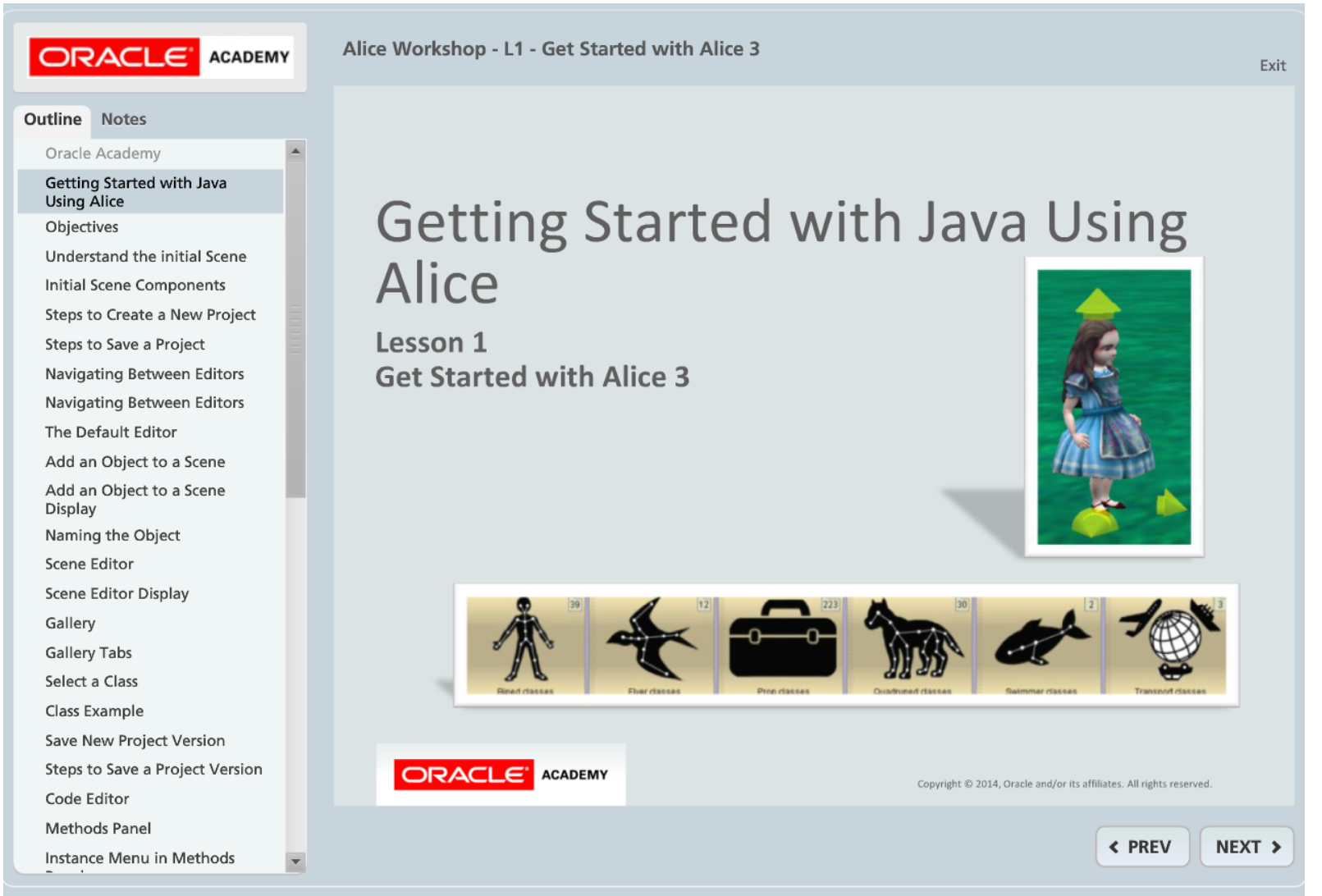
Creating Java Programs with Greenfoot

This workshop engages students who understand basic programming concepts to create 2-D games using Java. Students will learn detailed object-oriented programming terminology and concepts while creating 2-D games in a fun and interactive environment.

Duration 16 Hours

ORACLE

Plan de formación y certificación en Java



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

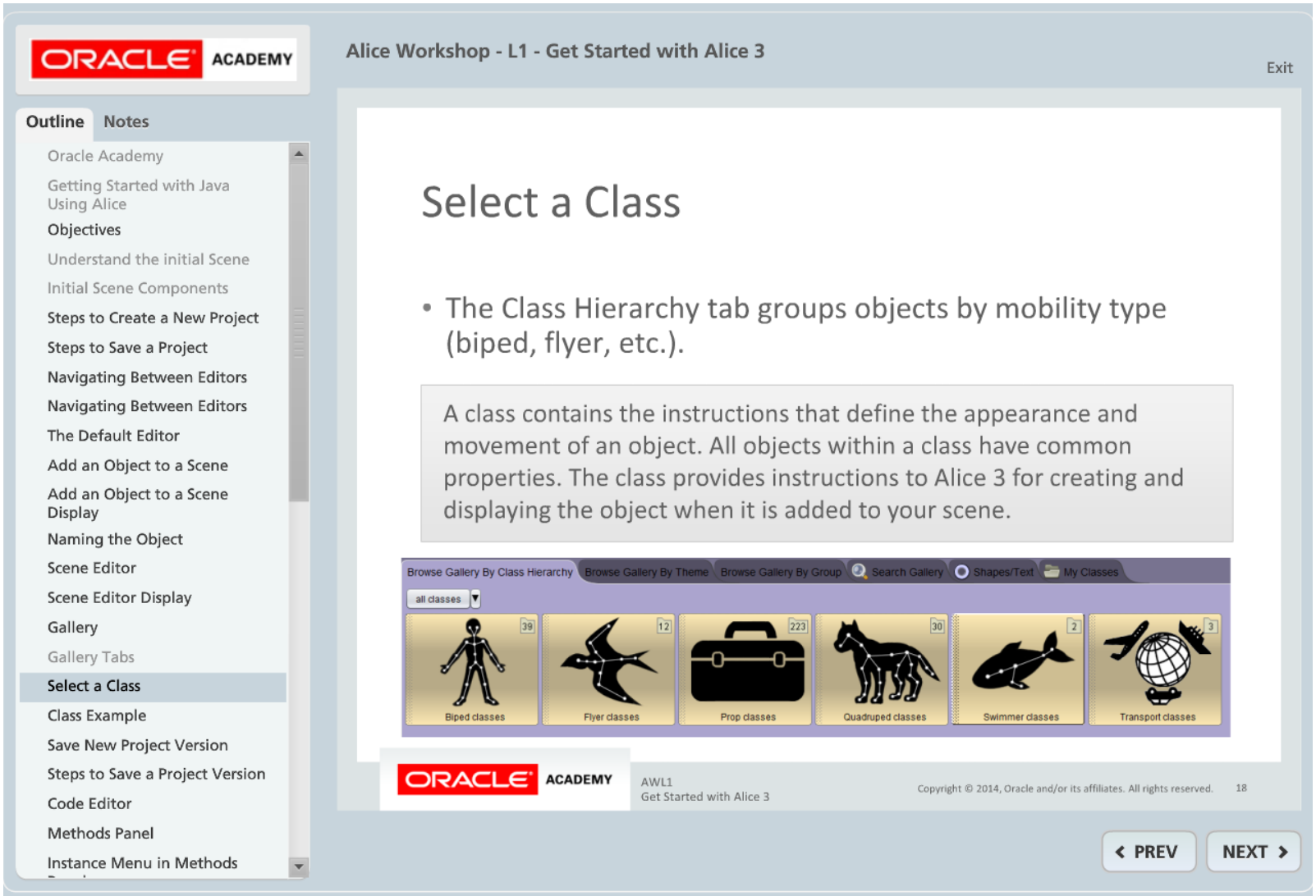
#PUEDAY16

Plan de formación y certificación en Java

pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

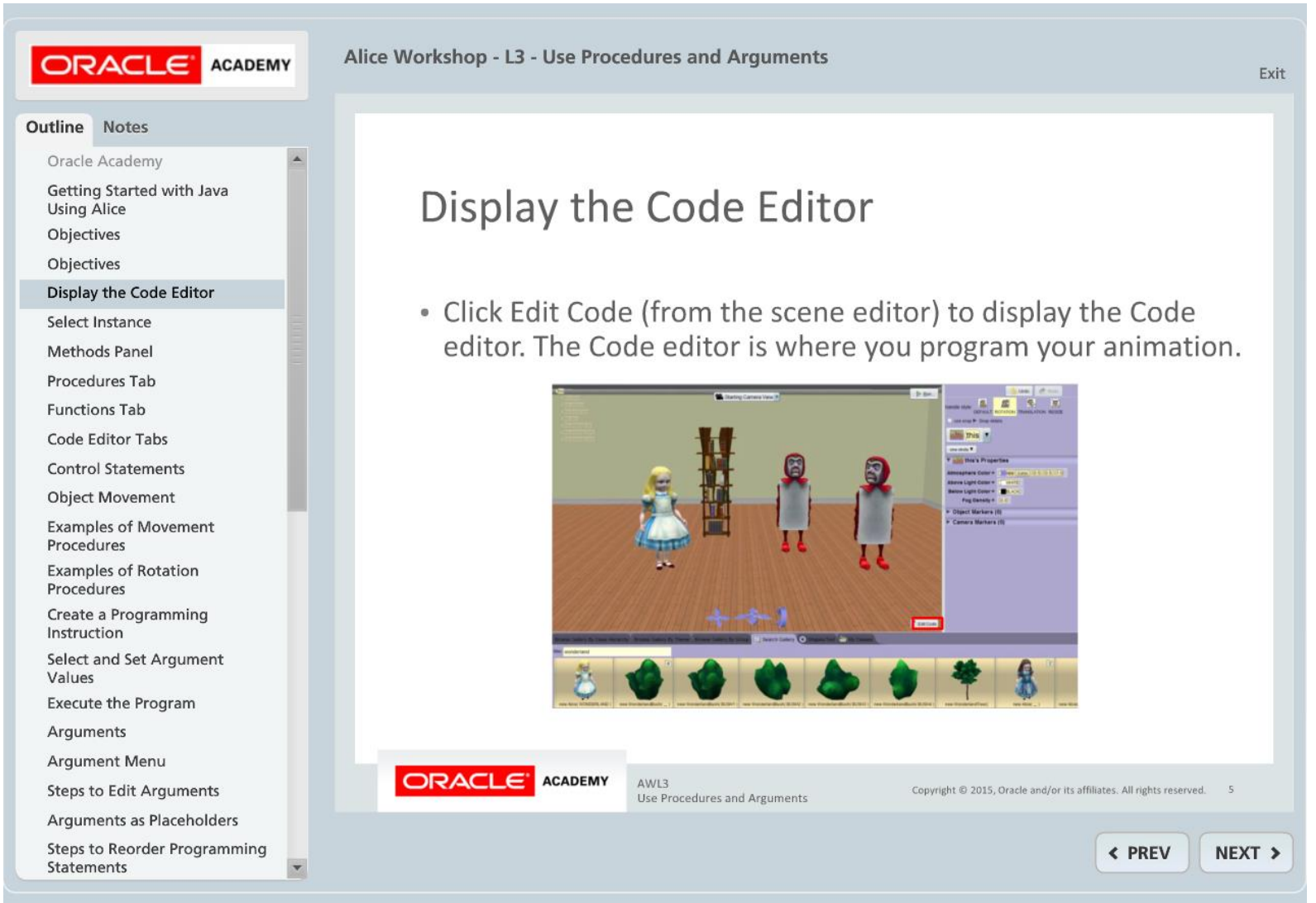
Plan de formación y certificación en Java



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Plan de formación y certificación en Java



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

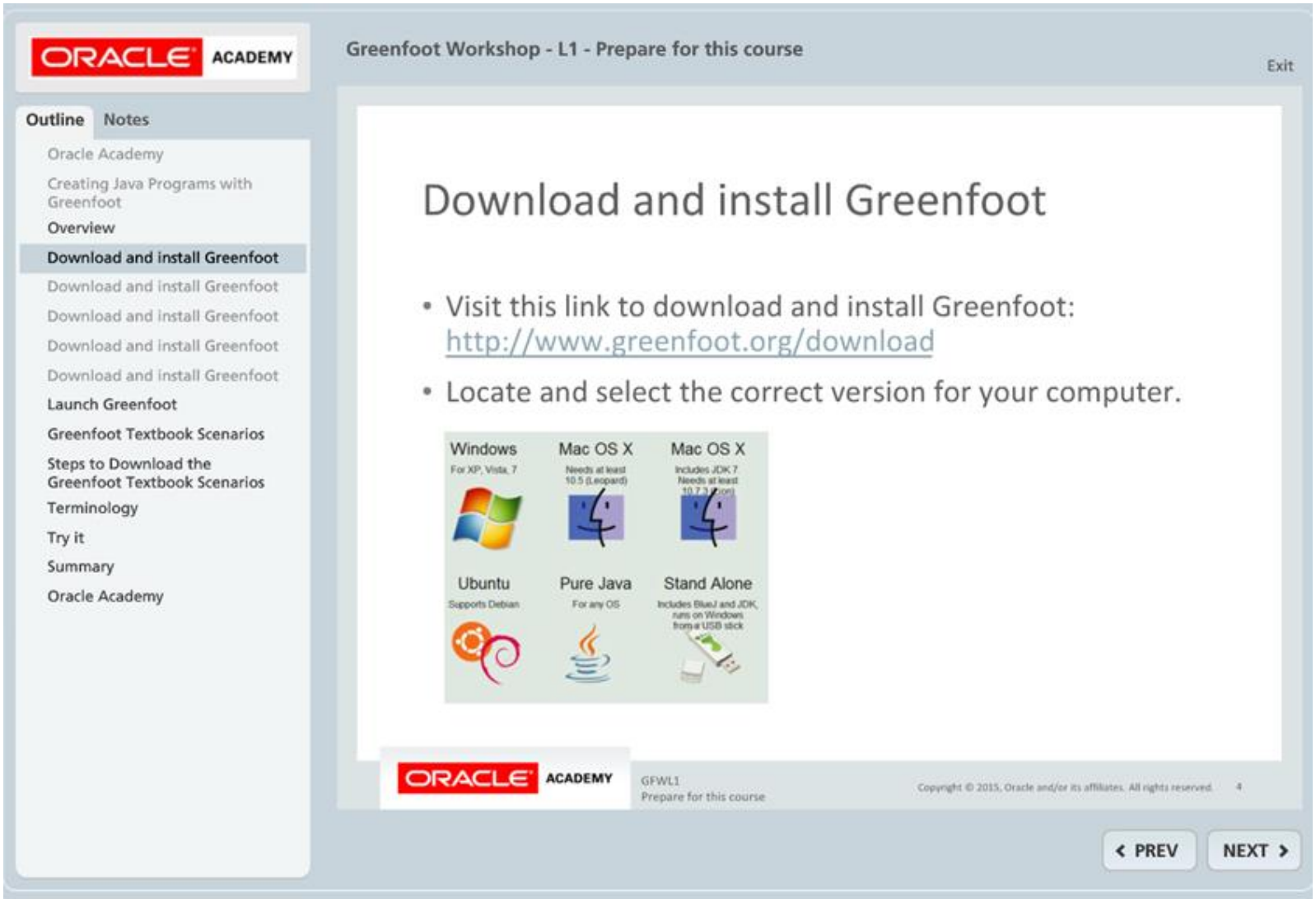
#PUEDAY16

Plan de formación y certificación en Java

pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Plan de formación y certificación en Java



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Plan de formación y certificación en Java

ORACLE ACADEMY

Greenfoot Workshop - L3 - Using Methods, Variables, and Parameters

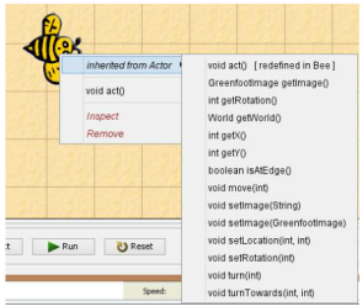
Exit

Outline | Notes

- Oracle Academy
- Creating Java Programs with Greenfoot
- Overview
- Methods Example
- Methods
- Inheritance
- View Inherited Methods in Object Menu
- Steps to View Inherited Methods in the Code Editor
- Method Summary
- Method Components
- Method Signature
- Return Types
- Methods with Void Return Types
- Invoking Methods with Void Return Types
- Ask Objects Questions
- Methods with Non-void Return Types
- Examples of Non-Void Return Types
- Method Parameters
- Examples of Method Parameters
- Method Parameter Lists
- Object Properties

View Inherited Methods in Object Menu

- The object menu displays all of the methods that the instance inherits from its class and superclass.
  - Right click on the instance to display the menu.
  - Inherited From Actor displays a list of the methods that the class inherits from the Actor superclass.



ORACLE ACADEMY

GFWL3
Using Methods, Variables, and Parameters

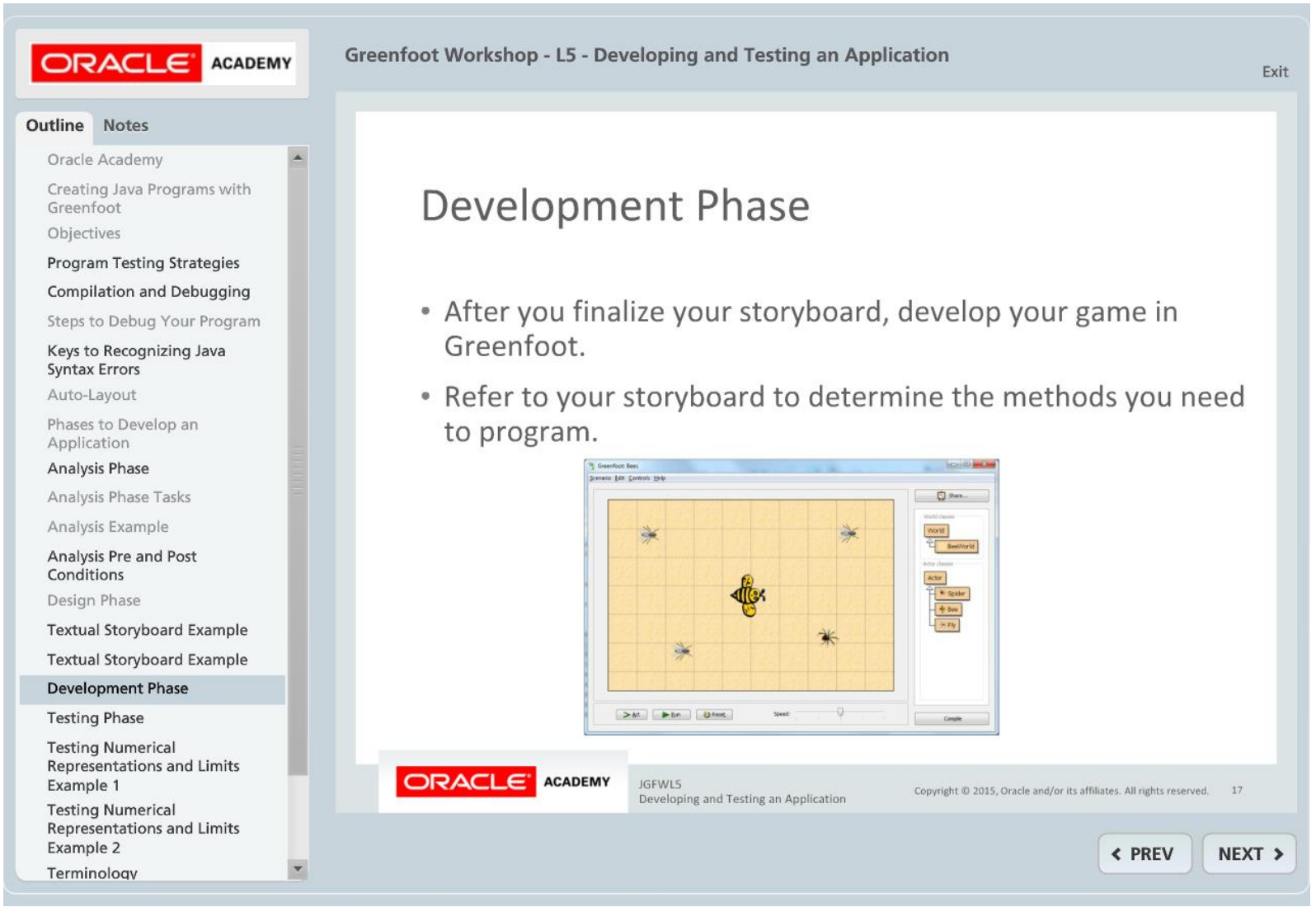
7

PREV NEXT

pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Plan de formación y certificación en Java



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Plan de formación y certificación en Java

ORACLE
Certified Professional

Oracle Certified Professional
Java SE Programming

Curso:
Java Programming

ORACLE
Certified Associate

Oracle Certified Associate
Java SE Programming

Curso:
Java Fundamentals

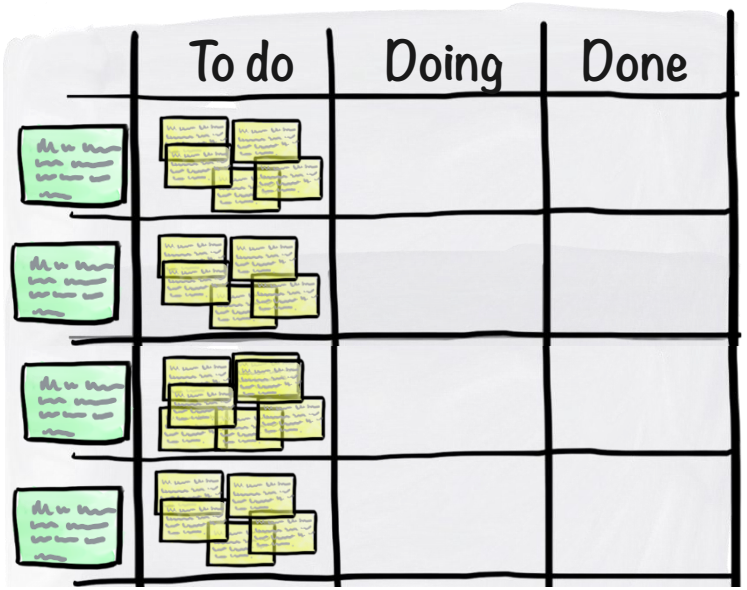
ORACLE
Certified Associate

Java Foundations Certified
Junior Associate

Curso:
Java Foundations

Creating Java Programs
with Greenfot

Getting started with Java
using Alice

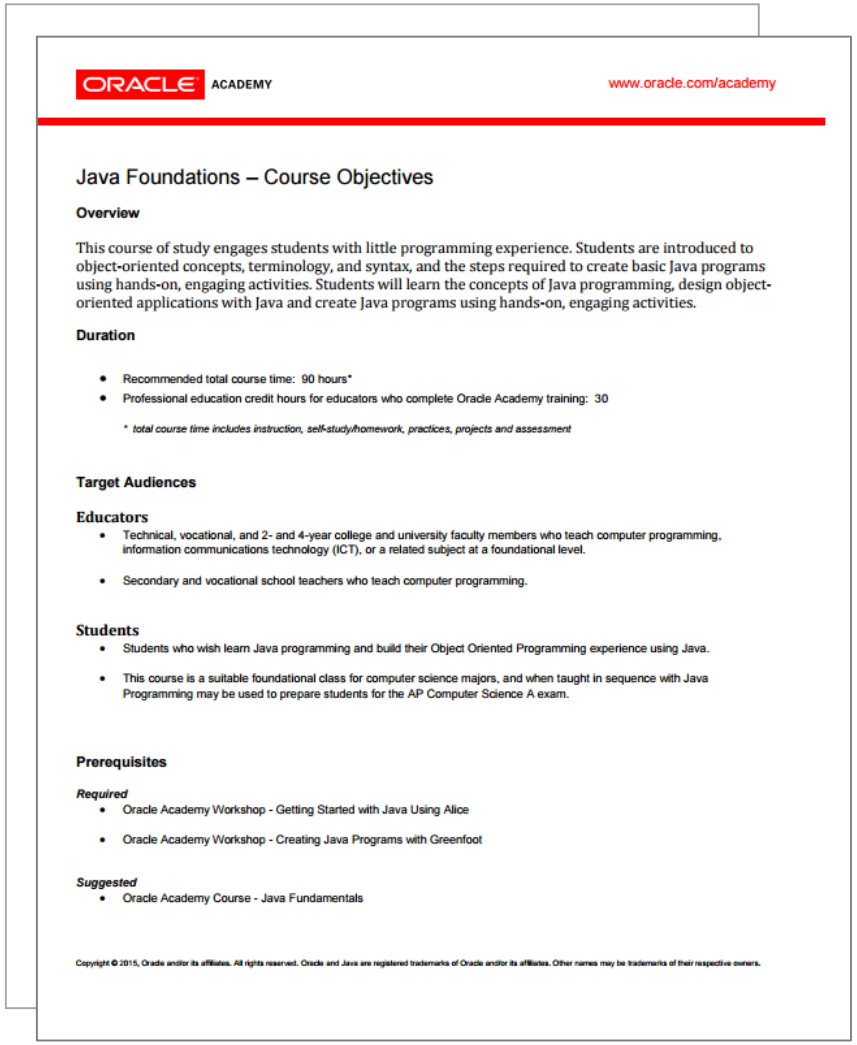


Kanban Board
(Visual Management)

Agenda

1.- Introducción
2.- Plan de formación y certificación en Java
3.- Curso Java Foundations
4.- Estructura del curso Java Foundations
5.- Demo: Curso Java Foundations
6.- Oracle iLearning: Informes y diplomas

Curso Java Foundations



ORACLE ACADEMY www.oracle.com/academy

Java Foundations – Course Objectives

Overview

This course of study engages students with little programming experience. Students are introduced to object-oriented concepts, terminology, and syntax, and the steps required to create basic Java programs using hands-on, engaging activities. Students will learn the concepts of Java programming, design object-oriented applications with Java and create Java programs using hands-on, engaging activities.

Duration

- Recommended total course time: 90 hours*
- Professional education credit hours for educators who complete Oracle Academy training: 30

** total course time includes instruction, self-study/homework, practices, projects and assessment*

Target Audiences

Educators

- Technical, vocational, and 2- and 4-year college and university faculty members who teach computer programming, information communications technology (ICT), or a related subject at a foundational level.
- Secondary and vocational school teachers who teach computer programming.

Students

- Students who wish learn Java programming and build their Object Oriented Programming experience using Java.
- This course is a suitable foundational class for computer science majors, and when taught in sequence with Java Programming may be used to prepare students for the AP Computer Science A exam.

Prerequisites

Required

- Oracle Academy Workshop - Getting Started with Java Using Alice
- Oracle Academy Workshop - Creating Java Programs with Greenfoot

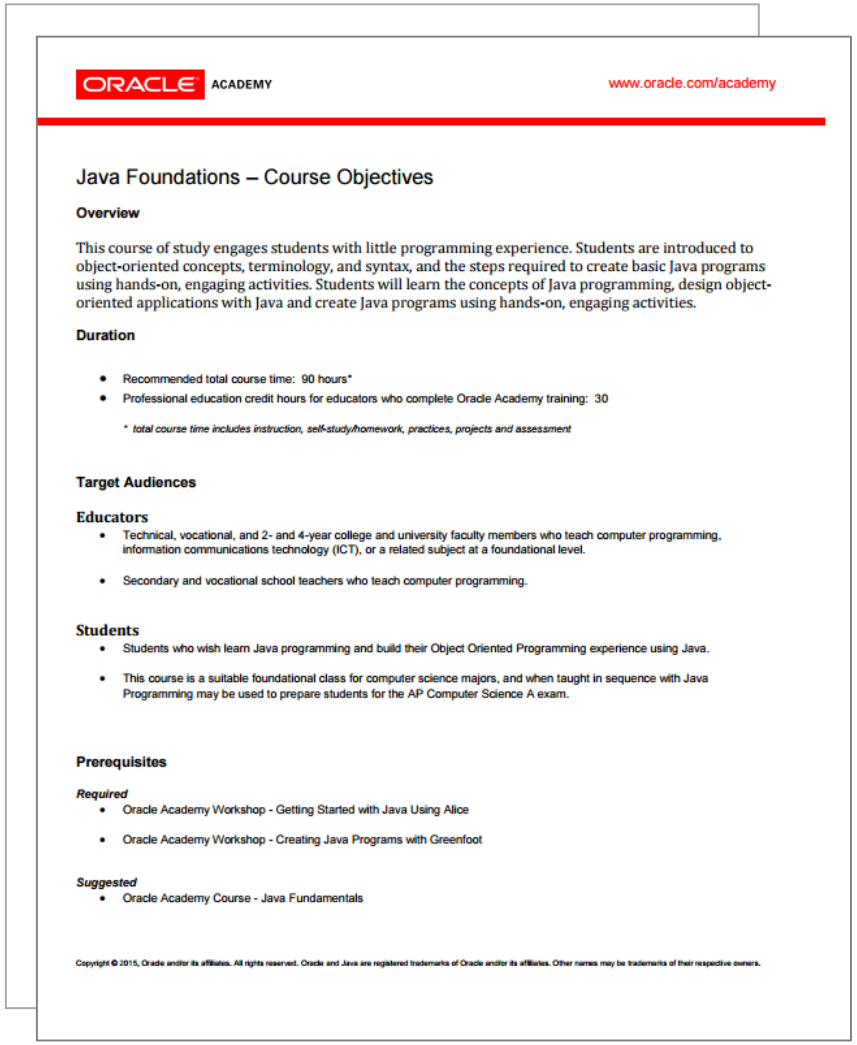
Suggested

- Oracle Academy Course - Java Fundamentals

Descripción del curso

- Dirigido a los estudiantes con **poca o ninguna experiencia** en programación.
- Los estudiantes aprenderán la **sintaxis** y los **conceptos básicos de programación** en Java, así como a **diseñar sus primeras aplicaciones orientadas a objetos** dentro la plataforma Java.
- Ofrece gran variedad de **actividades**, **tests**, **juegos** y **prácticas** que fomentan la participación y aprendizaje de los estudiantes.
- Ofrece una **base de conocimiento sólida y robusta** en la plataforma Java y el paradigma OOP. Permite comenzar a coger experiencia en el uso del API nativo de Java.
- Enseña conceptos fundamentales en el desarrollo de software, además de buenas prácticas, como ***clean code*** y ***refactorización***.

Curso Java Foundations



ORACLE ACADEMY www.oracle.com/academy

Java Foundations – Course Objectives

Overview

This course of study engages students with little programming experience. Students are introduced to object-oriented concepts, terminology, and syntax, and the steps required to create basic Java programs using hands-on, engaging activities. Students will learn the concepts of Java programming, design object-oriented applications with Java and create Java programs using hands-on, engaging activities.

Duration

- Recommended total course time: 90 hours*
- Professional education credit hours for educators who complete Oracle Academy training: 30

** total course time includes instruction, self-study/homework, practices, projects and assessment*

Target Audiences

Educators

- Technical, vocational, and 2- and 4-year college and university faculty members who teach computer programming, information communications technology (ICT), or a related subject at a foundational level.
- Secondary and vocational school teachers who teach computer programming.

Students

- Students who wish learn Java programming and build their Object Oriented Programming experience using Java.
- This course is a suitable foundational class for computer science majors, and when taught in sequence with Java Programming may be used to prepare students for the AP Computer Science A exam.

Prerequisites

Required

- Oracle Academy Workshop - Getting Started with Java Using Alice
- Oracle Academy Workshop - Creating Java Programs with Greenfoot

Suggested

- Oracle Academy Course - Java Fundamentals

Contenido del curso

Section 1 – Introduction

- 1.1 About the Course
- 1.2 A Brief History
- 1.3 Setting up Java

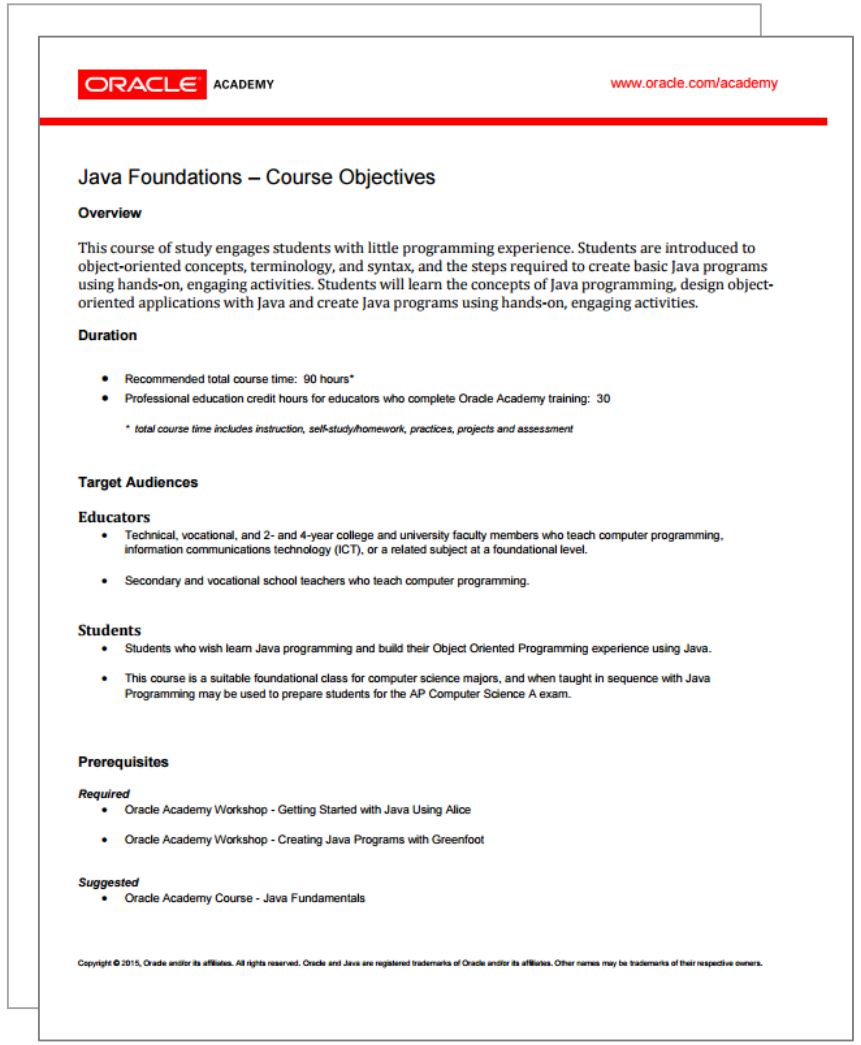
Section 2 – Java Software Development

- 2.1 The Software Development Process
- 2.2 What is my Program Doing?
- 2.3 Introduction to Object-Oriented Programming Concepts

Section 3 – Java Data Types

- 3.1 What is a Variable?
- 3.2 Numeric Data
- 3.3 Textual Data
- 3.4 Converting Between Data Types
- 3.5 Keyboard Input

Curso Java Foundations



ORACLE ACADEMY www.oracle.com/academy

Java Foundations – Course Objectives

Overview

This course of study engages students with little programming experience. Students are introduced to object-oriented concepts, terminology, and syntax, and the steps required to create basic Java programs using hands-on, engaging activities. Students will learn the concepts of Java programming, design object-oriented applications with Java and create Java programs using hands-on, engaging activities.

Duration

- Recommended total course time: 90 hours*
- Professional education credit hours for educators who complete Oracle Academy training: 30

** total course time includes instruction, self-study/homework, practices, projects and assessment*

Target Audiences

Educators

- Technical, vocational, and 2- and 4-year college and university faculty members who teach computer programming, information communications technology (ICT), or a related subject at a foundational level.
- Secondary and vocational school teachers who teach computer programming.

Students

- Students who wish learn Java programming and build their Object Oriented Programming experience using Java.
- This course is a suitable foundational class for computer science majors, and when taught in sequence with Java Programming may be used to prepare students for the AP Computer Science A exam.

Prerequisites

Required

- Oracle Academy Workshop - Getting Started with Java Using Alice
- Oracle Academy Workshop - Creating Java Programs with Greenfoot

Suggested

- Oracle Academy Course - Java Fundamentals

Contenido del curso

Section 4 – Java Methods and Library Classes

- 4.1 What is a Method?
- 4.2 The import Declaration and Packages
- 4.3 The String Class
- 4.4 The Random Class
- 4.5 The Math Class

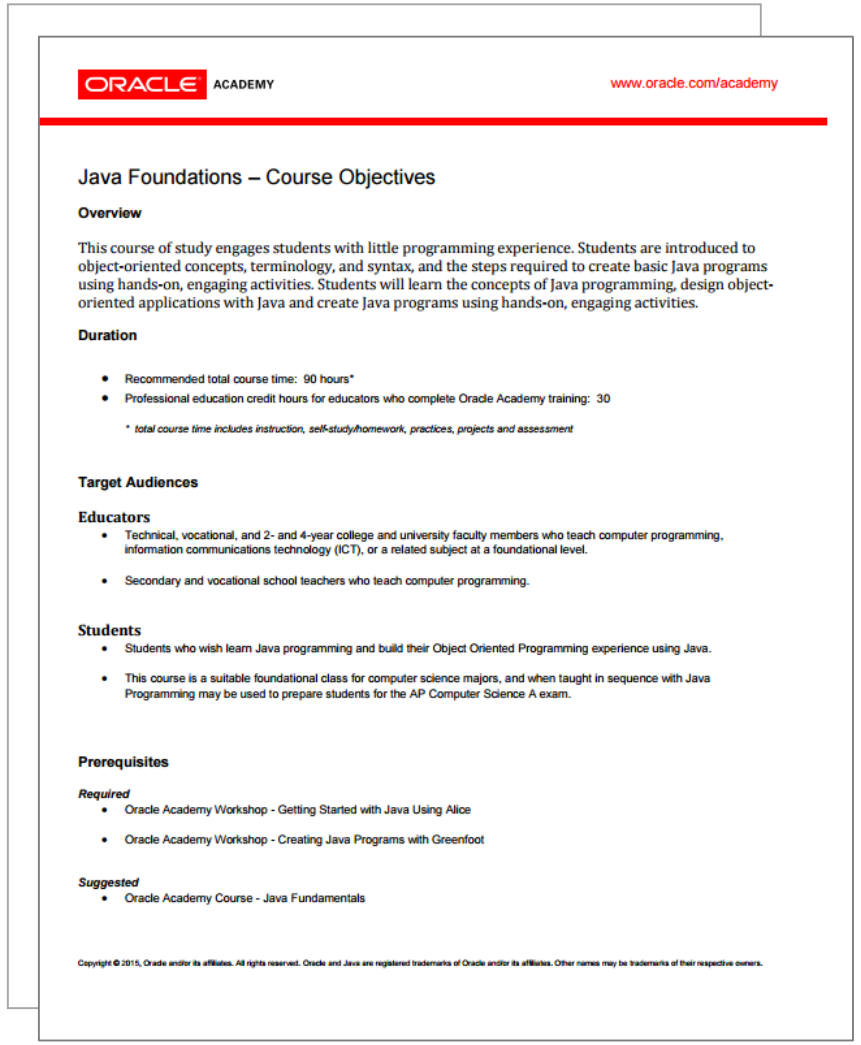
Section 5 – Decision Statements

- 5.1 Boolean Expressions and if/else Constructs
- 5.2 Understanding Conditional Execution
- 5.3 switch Statement

Section 6 – Loop Constructs

- 6.1 for Loops
- 6.2 while and do-while Loops
- 6.3 Using break and continue Statements

Curso Java Foundations



ORACLE ACADEMY www.oracle.com/academy

Java Foundations – Course Objectives

Overview

This course of study engages students with little programming experience. Students are introduced to object-oriented concepts, terminology, and syntax, and the steps required to create basic Java programs using hands-on, engaging activities. Students will learn the concepts of Java programming, design object-oriented applications with Java and create Java programs using hands-on, engaging activities.

Duration

- Recommended total course time: 90 hours*
- Professional education credit hours for educators who complete Oracle Academy training: 30

** total course time includes instruction, self-study/homework, practices, projects and assessment*

Target Audiences

Educators

- Technical, vocational, and 2- and 4-year college and university faculty members who teach computer programming, information communications technology (ICT), or a related subject at a foundational level.
- Secondary and vocational school teachers who teach computer programming.

Students

- Students who wish learn Java programming and build their Object Oriented Programming experience using Java.
- This course is a suitable foundational class for computer science majors, and when taught in sequence with Java Programming may be used to prepare students for the AP Computer Science A exam.

Prerequisites

Required

- Oracle Academy Workshop - Getting Started with Java Using Alice
- Oracle Academy Workshop - Creating Java Programs with Greenfoot

Suggested

- Oracle Academy Course - Java Fundamentals

Contenido del curso

Section7– Creating Classes

- 7.1 Creating a Class
- 7.2 Instantiating Objects
- 7.3 Constructors
- 7.4 Overloading Methods
- 7.5 Object Interaction and Encapsulation
- 7.6 Static Variables and Methods

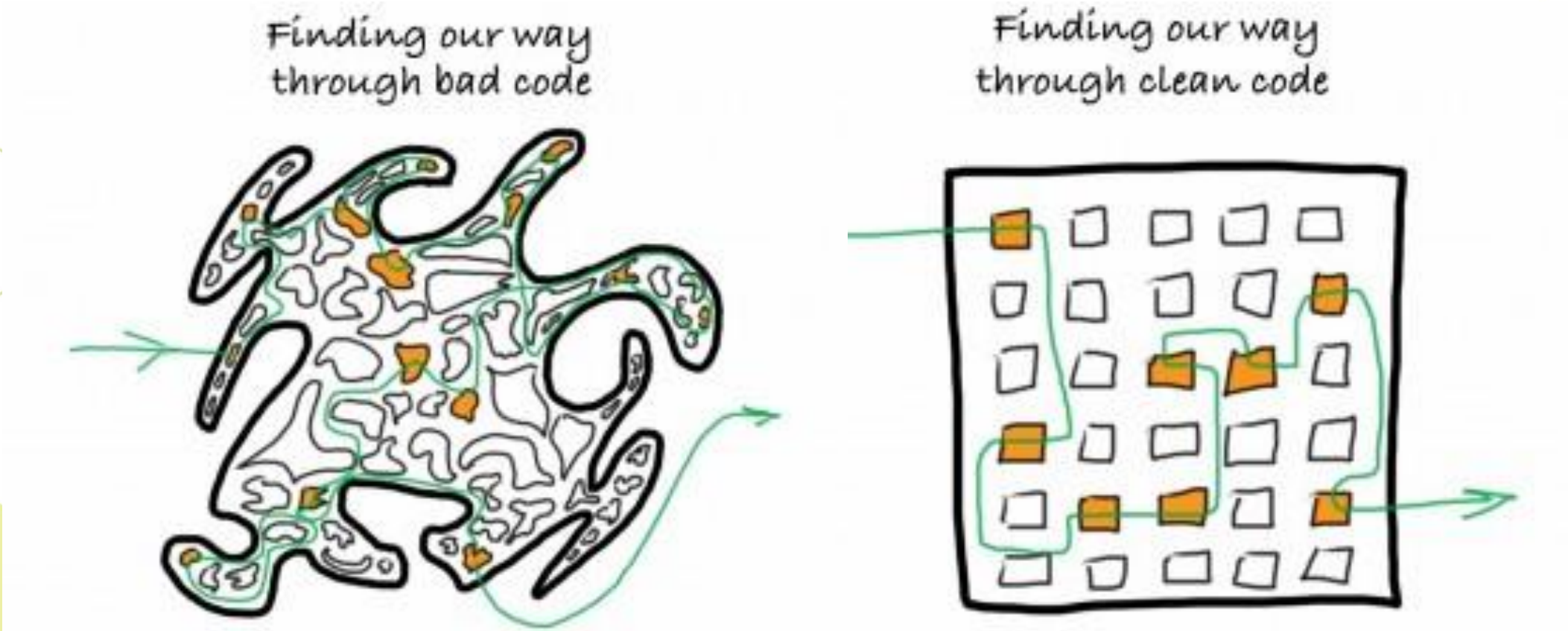
Section 8 – Arrays and Exceptions

- 8.1 One-dimensional Arrays
- 8.2 ArrayLists
- 8.3 Exception Handling
- 8.4 Debugging Concepts and Techniques

Section 9 – JavaFX

- 9.1 Introduction to Java FX
- 9.2 Colors and Shapes
- 9.3 Graphics, Audio and MouseEvents

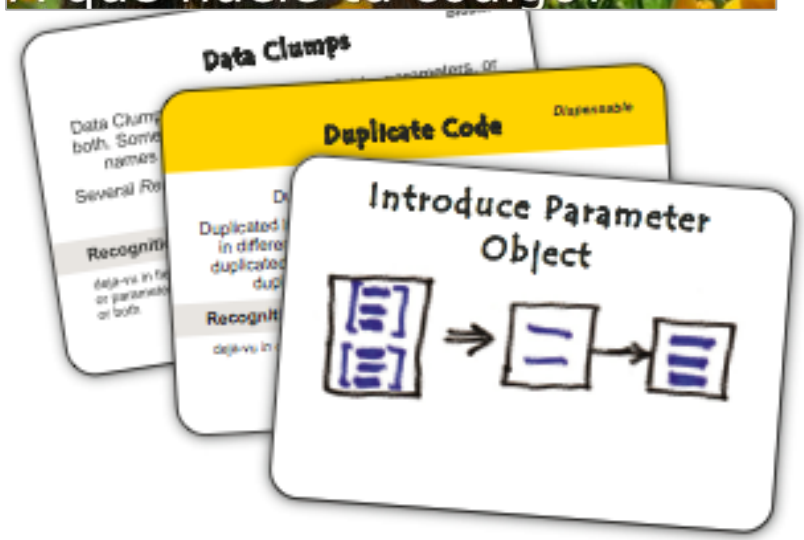
Curso Java Foundations

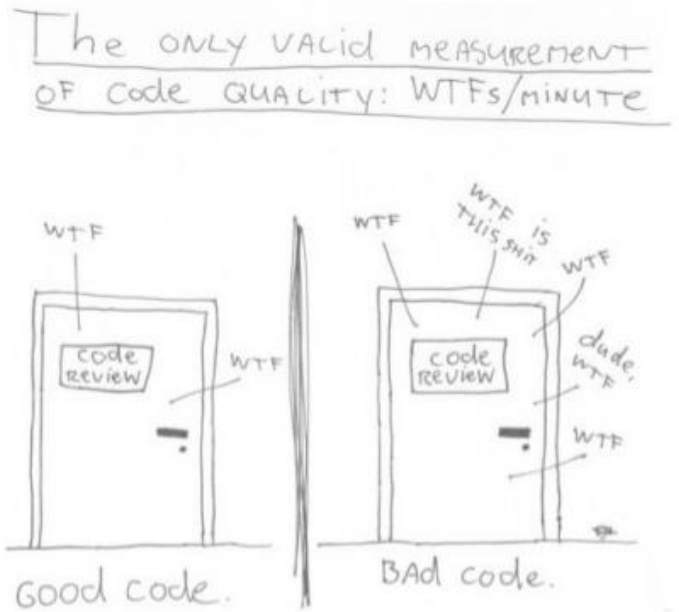


"Any fool can write code that a computer can understand.
Good programmers write code that humans can understand."

Martin Fowler, 2008

Curso Java Foundations





pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Curso Java Foundations

"La Regla del Boy Scout"

"Always leave the campground cleaner than you found it."

"Try and leave this world a little better than you found it."

Curso Java Foundations

OCA: Java Foundations

(en exclusiva para centros Oracle Academy)

ORACLE

Certified Associate

Java Foundations Certified
Junior Associate

Examen 1Z0-811:
Java Foundations

1Z0-811: Java Foundations

- **Plataforma:** Java SE 8
- **Duración:** 150 minutos
- **Núm. Preguntas:** 75 preguntas
- **Nota aprobado:** 65 %
- **Precio:** ~~8~~ → 61,5 €

Contenidos

What is Java?
Java Basics
Basic Java Elements
Working with Java Data Types
Working with Java Operator
Working with the String Class
Working with the Random and Match Classes
Using Decision Statements
Using Looping Statements
Debugging and Exception Handling
Arrays and ArrayLists
Classes and Constructors
Java Methods

25%

de descuento
en certificaciones oficiales de Oracle

pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Curso Java Foundations

Given the code fragment:

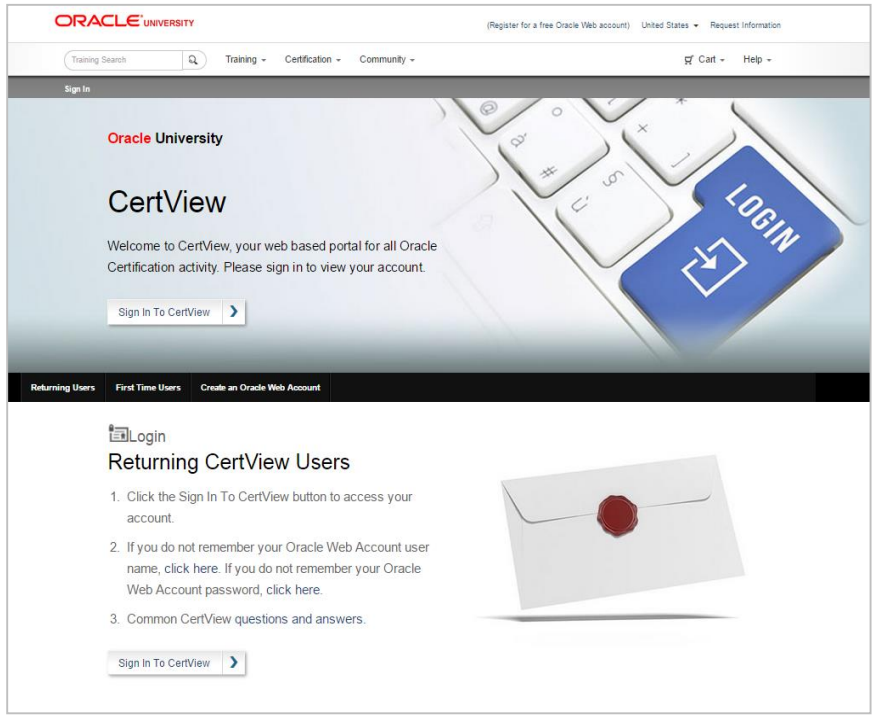
```
String a = “Java”;
String b = new String(“Java”);
Sytemm.out.println(a.equals(b));
System.out.println(a==b);
```

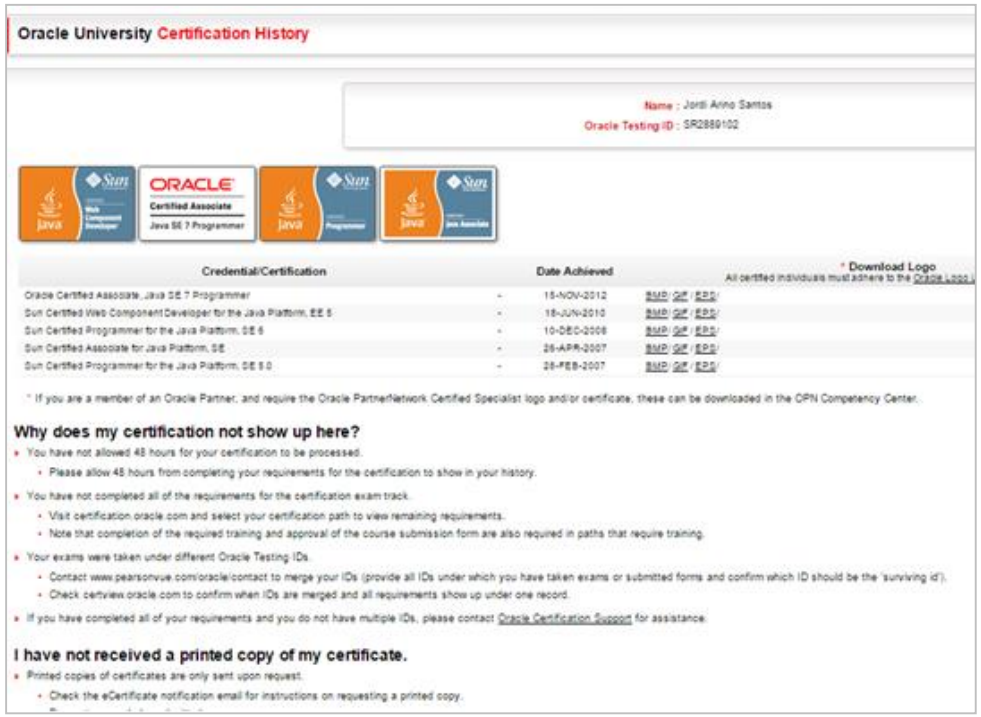
What is the result?

A) false
   false
B) true
   true
C) false
   true
D) frue
   false

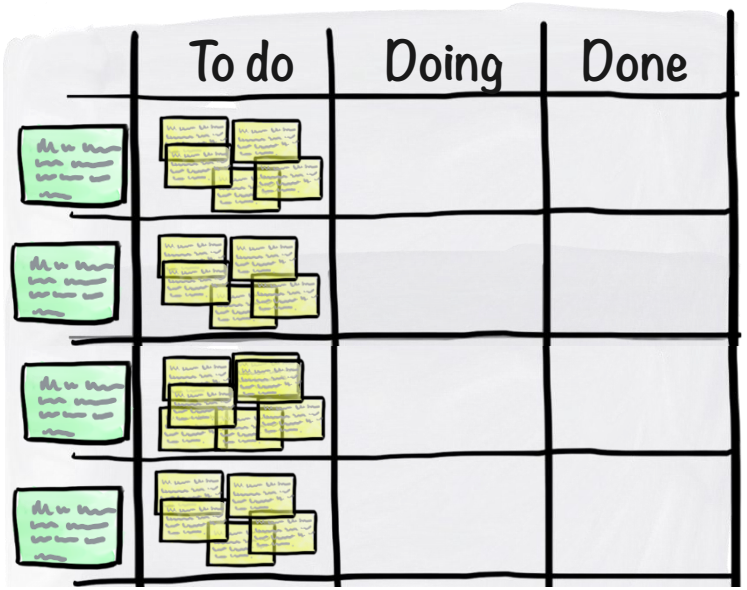
Curso Java Foundations

Oracle CertView





https://certview.oracle.com



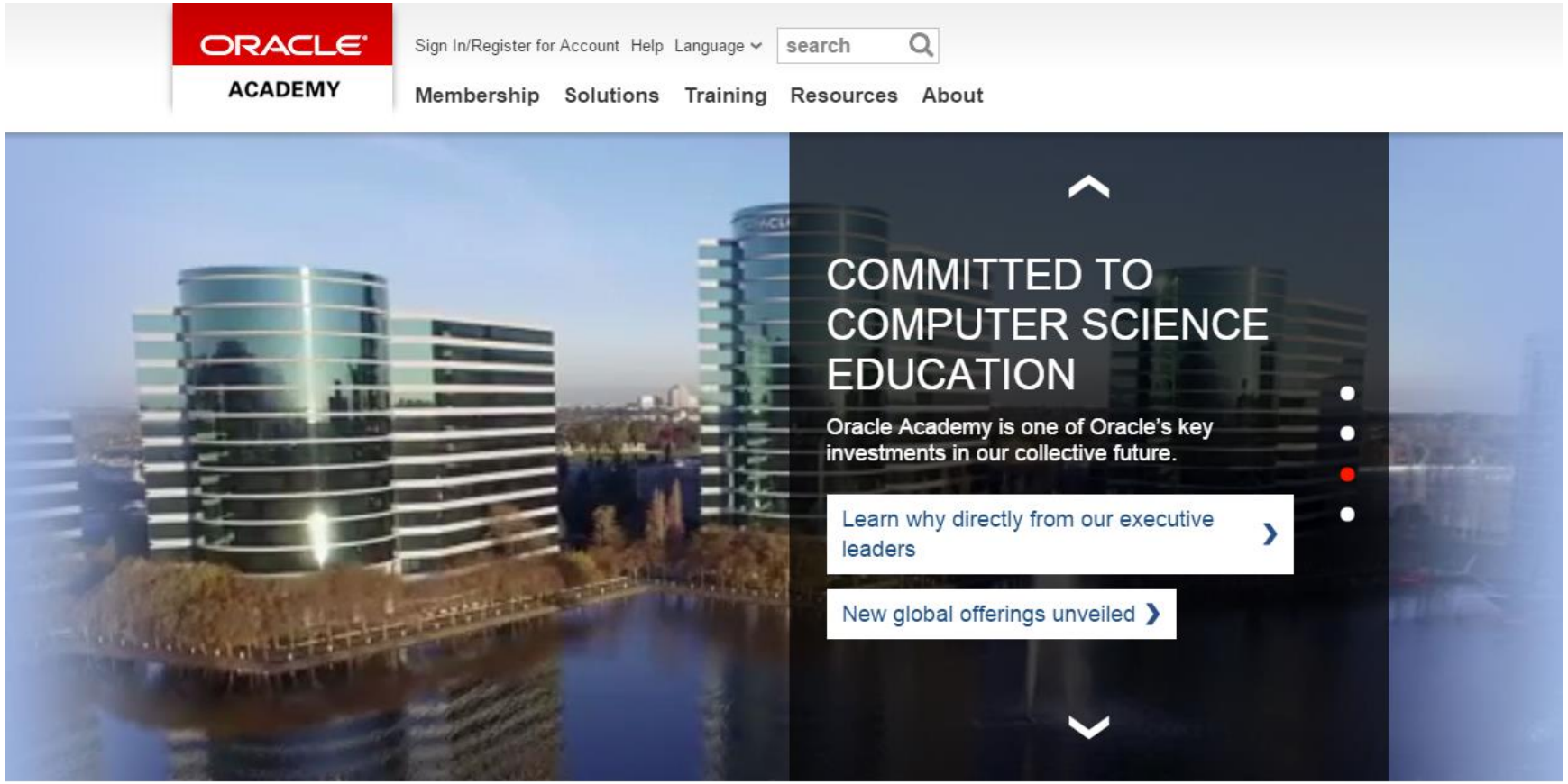
Kanban Board
(Visual Management)

Agenda

1.- Introducción
2.- Plan de formación y certificación en Java
3.- Curso Java Foundations
4.- Estructura del curso Java Foundations
5.- Demo: Curso Java Foundations
6.- Oracle iLearning: Informes y diplomas

Estructura del curso Java Foundations

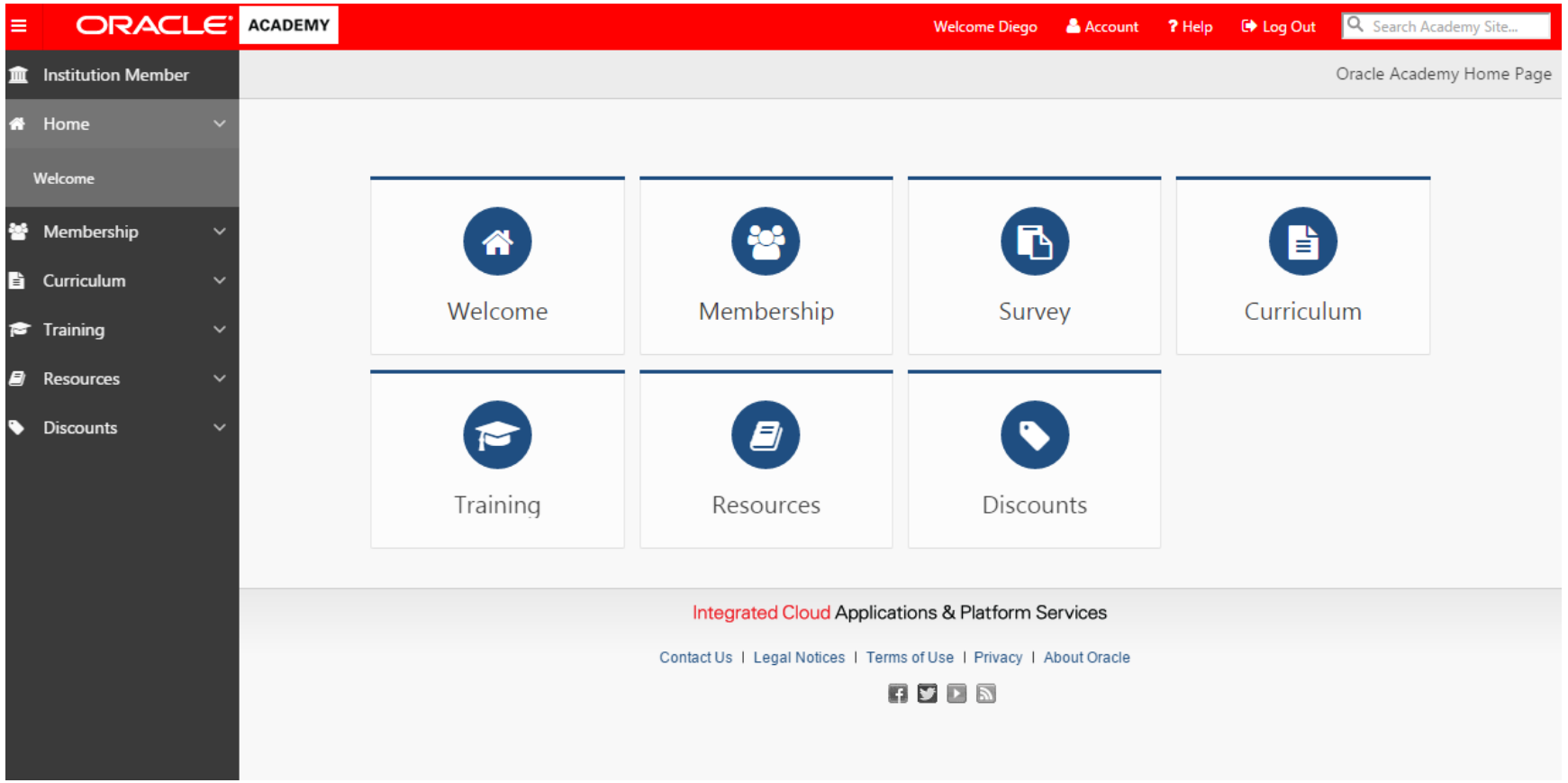
Nueva plataforma Oracle Academy 3.0



https://academy.oracle.com

Estructura del curso Java Foundations

Nueva plataforma Oracle Academy 3.0



https://academy.oracle.com

Estructura del curso Java Foundations

http://ilearning.oracle.com/ilearn/en/learner/jsp/login.jsp?site=OracleAcad

Estructura del curso Java Foundations

Session Length 45 Minutes

	Session1	Session2	Session3	Session4	Session5
Week 1	Introduction				
Week 2	Java Software Development			Section 2 Quiz (15 preguntas)	
Week 3	Java Data Types			Section 3 Quiz A1-2 (15 preguntas)	
Week 4				Section 3 Quiz A3-5 (15 preguntas)	
Week 5	Java Methods and Library Classes			Section 4 Quiz A1-2 (15 preguntas)	
Week 6				Section 4 Quiz A3-5 (15 preguntas)	
Week 7	Decision Statements			Section 5 Quiz (15 preguntas)	
Week 8	Mid Term Exam Review			Mid Term Exam	
Week 9	Loop Constructs			Section 6 Quiz (15 preguntas)	
Week 10	Creating Classes			Section 7 Quiz A1-3 (15 preguntas)	
Week 11				Section 7 Quiz A4-6 (15 preguntas)	
Week 12	Arrays and Exceptions			Section 8 Quiz (15 preguntas)	
Week 13					
Week 14	Java FX			Section 9 Quiz (15 preguntas)	
Week 15					
Week 16	Final Exam Review			Final Exam	
Week 17	Final Project				
Week 18					

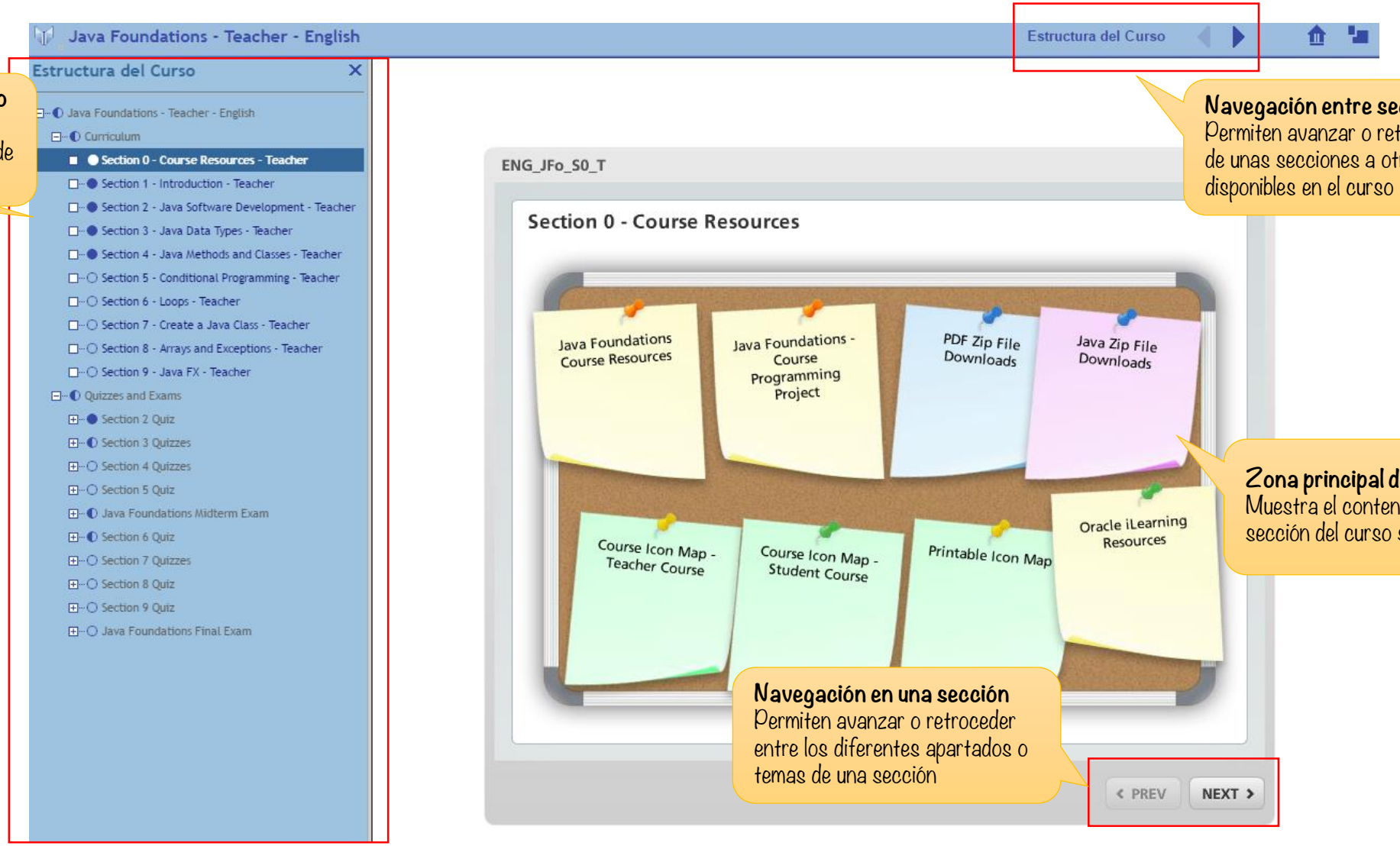
50 preguntas -> 3 intentos

50 preguntas -> 3 intentos

pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Estructura del curso Java Foundations



Estructura del curso
Panel de navegación por secciones y test de evaluación o quizzes

Navegación entre secciones
Permiten avanzar o retroceder de unas secciones a otras disponibles en el curso

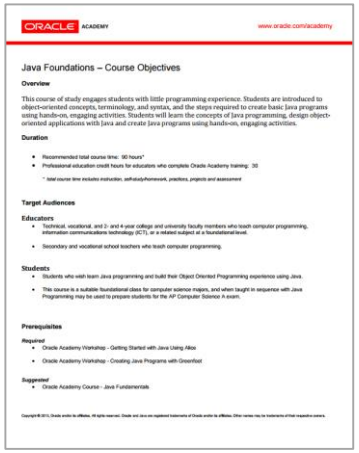
Zona principal de contenido
Muestra el contenido de la sección del curso seleccionada

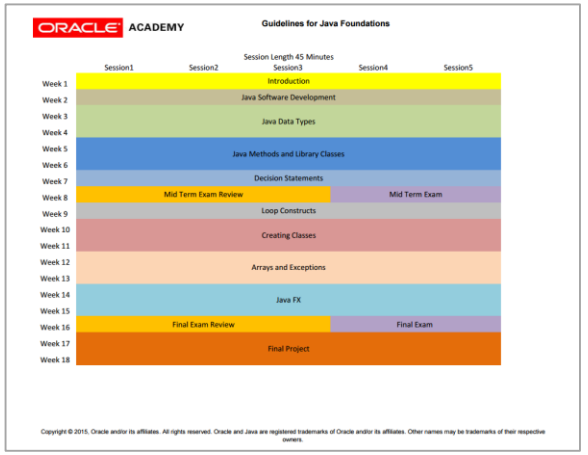
Navegación en una sección
Permiten avanzar o retroceder entre los diferentes apartados o temas de una sección

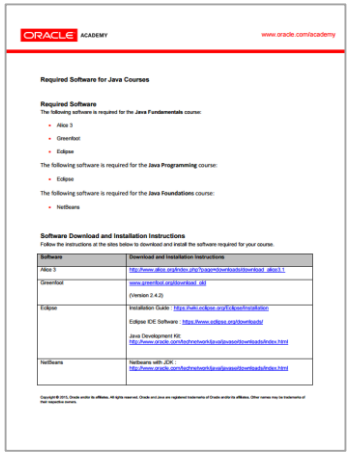
pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

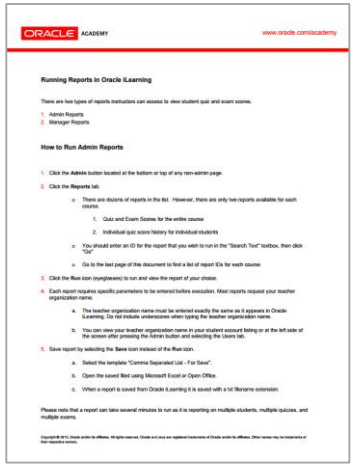
#PUEDAY16

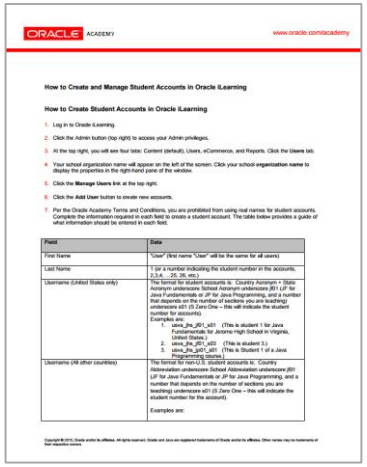
Estructura del curso Java Foundations







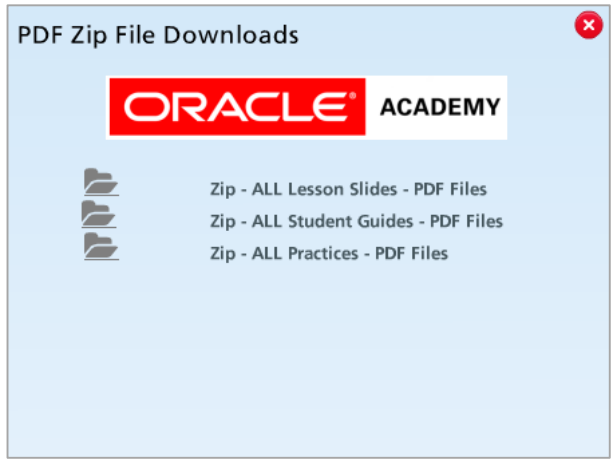


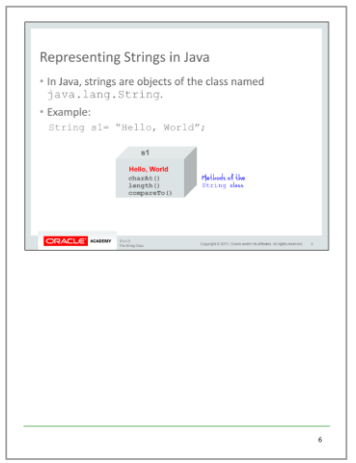


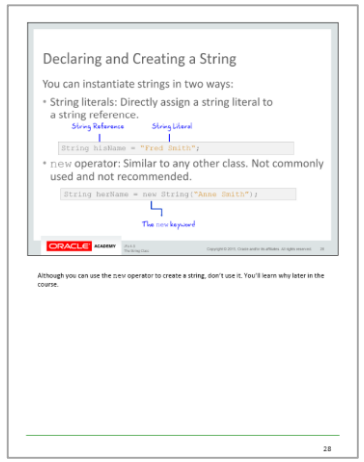
pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

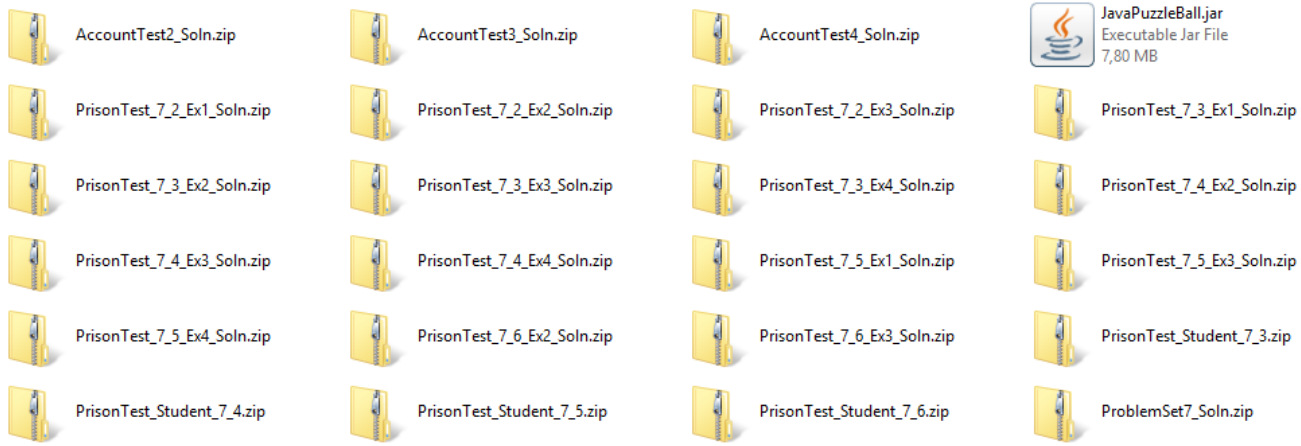
#PUEDAY16

Estructura del curso Java Foundations

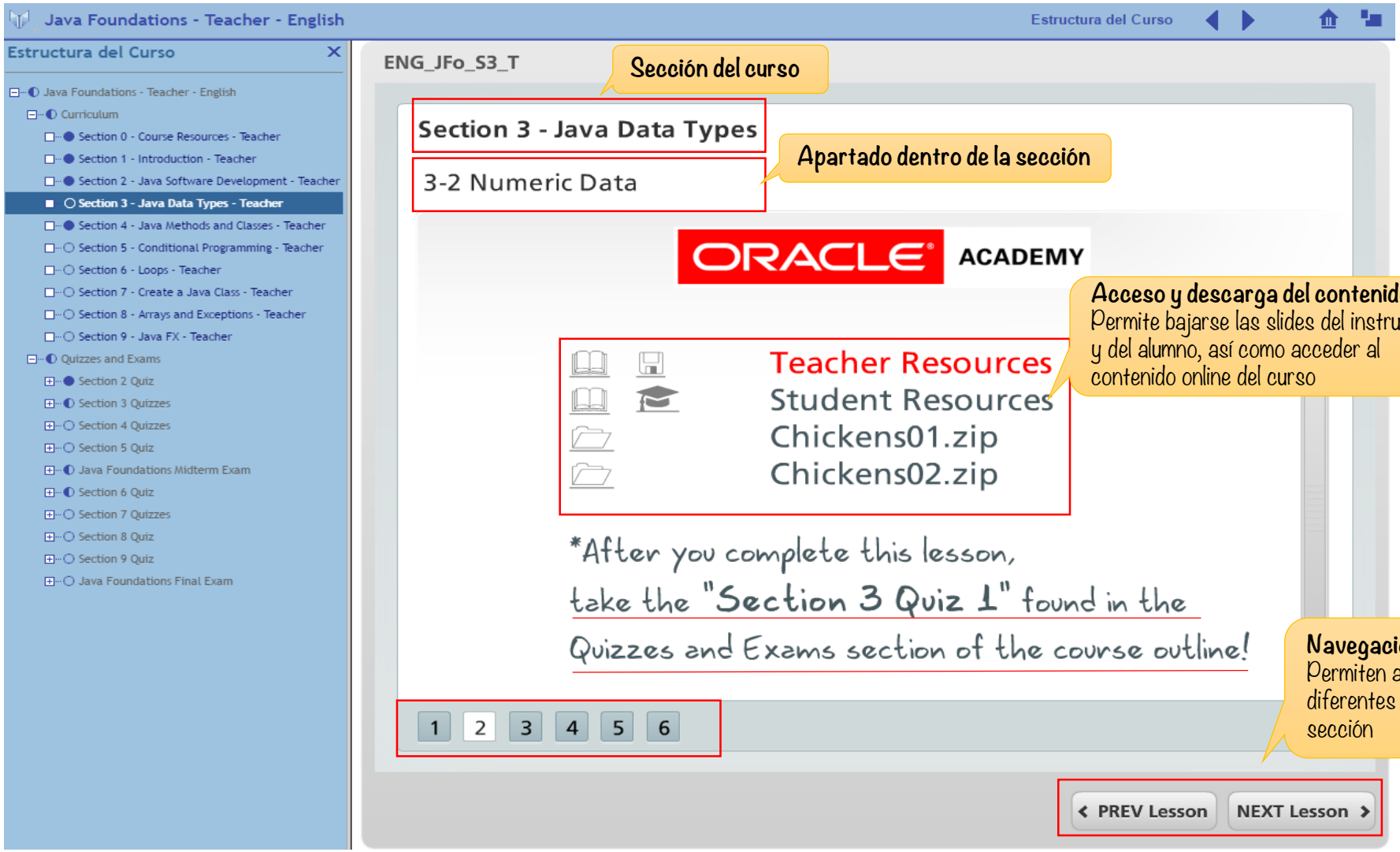




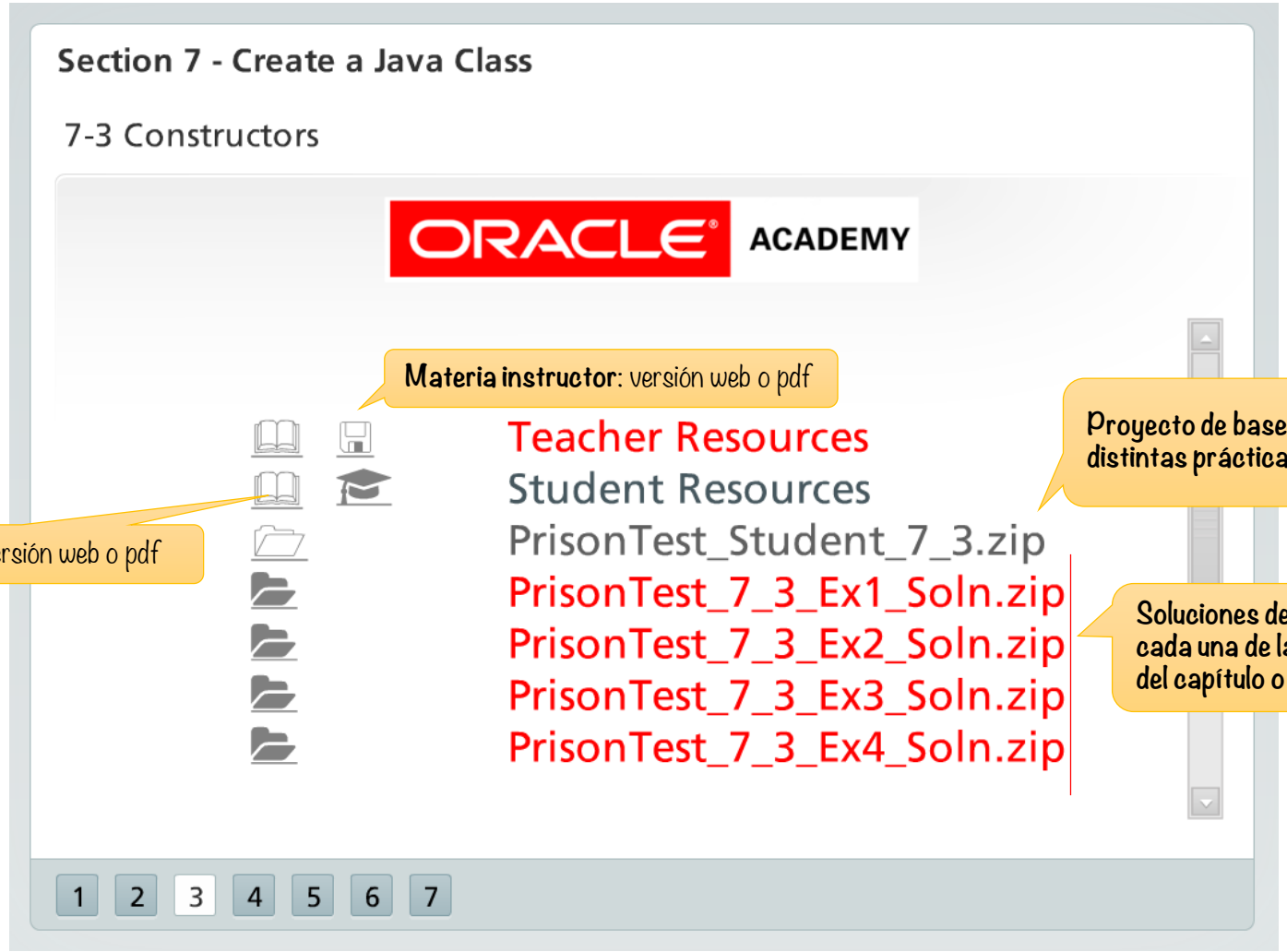




Estructura del curso Java Foundations



Estructura del curso Java Foundations



Materia alumno: versión web o pdf

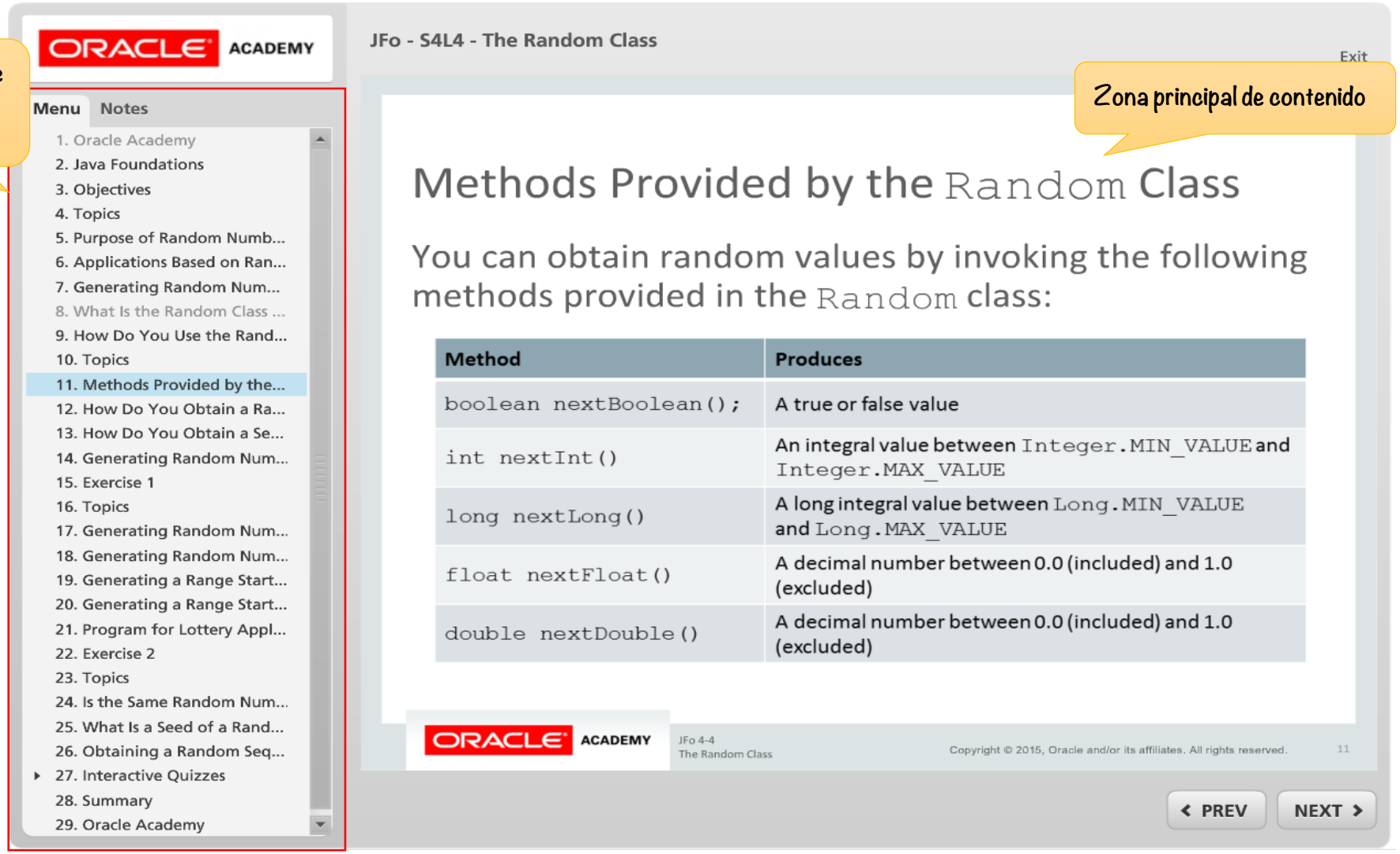
Proyecto de base para realizar las distintas prácticas guiadas

Soluciones de instructor para cada una de las prácticas guiadas del capítulo o apartado

Estructura del curso Java Foundations

Menú de navegación de un apartado de una sección del curso

Zona principal de contenido



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

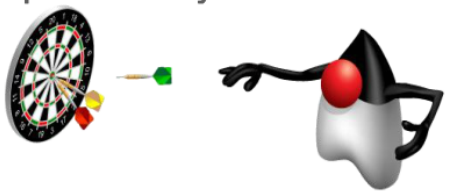
Estructura del curso Java Foundations

- Objetivos
- Sumario/Índice
- Contenido didáctico
- Ejercicios / Prácticas
- Resoluciones ejercicios
- Interactive Quizzes
- Resumen de contenido
- Resumen de la lección

Summary

In this lesson, you should have learned how to:

- Understand the memory consequences of instantiating objects
- Understand object references
- Understand the difference between stack and heap memory
- Understand how `Strings` are special objects



Estructura del curso Java Foundations

Match each item with a definition.

The process of turning source code (.java) into bytecode (.class).	Compiling
Readable by humans	Source code (.java)
Readable by the JRE	Bytecode (.class)

Interactive Quizzes

Estructura del curso Java Foundations

What would you type in line 5 to make this program print "Hello Java!" ?

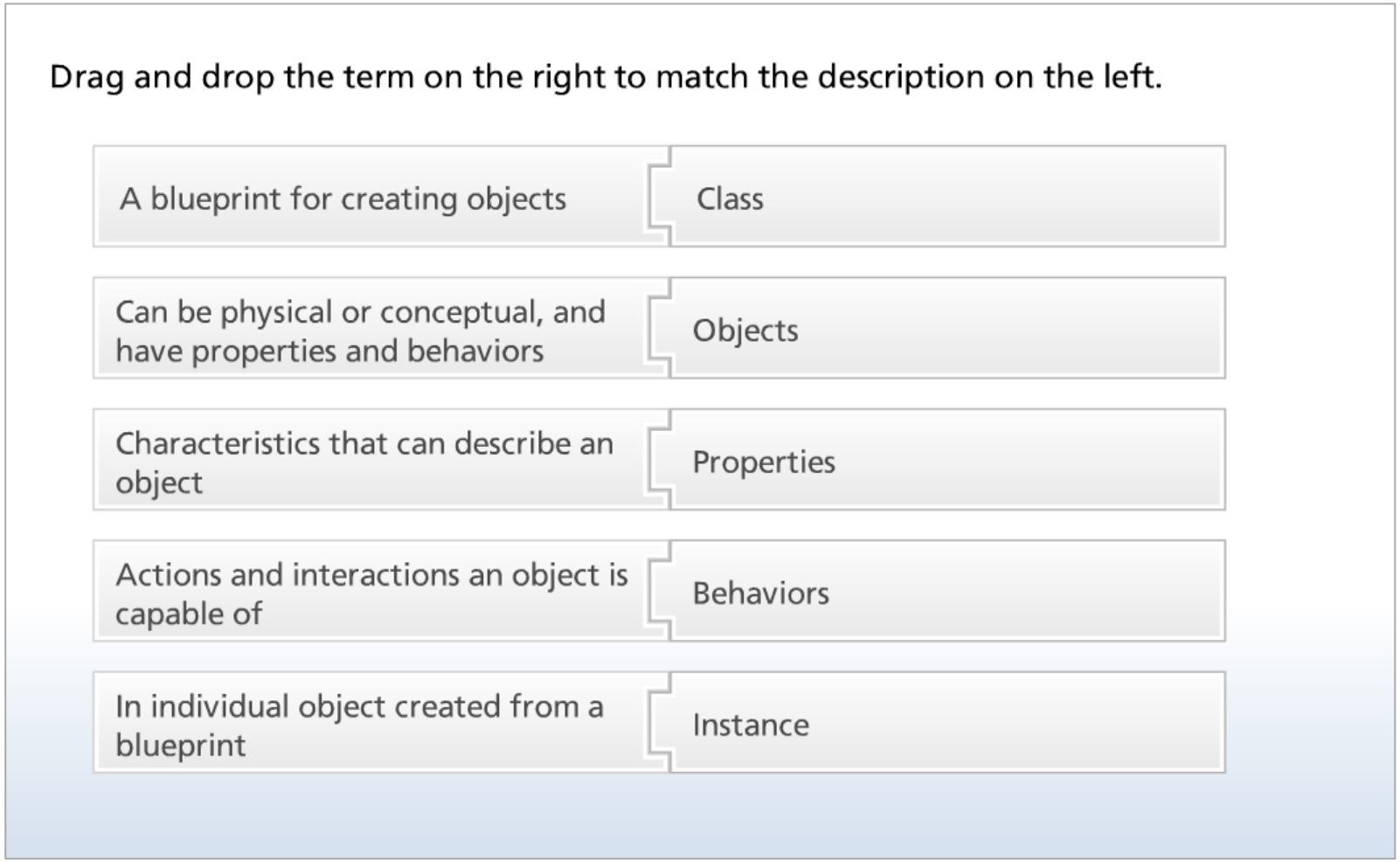
```
1
2  public class HelloWorld {
3
4      public static void main(String[] args) {
5          System.out.println("Hello Java!");
6      }
7  }
8
```

Interactive Quizzes

Estructura del curso Java Foundations

Interactive Quizzes

Estructura del curso Java Foundations



Interactive Quizzes

Estructura del curso Java Foundations

What should be the return type for this method, which returns a calculated value?

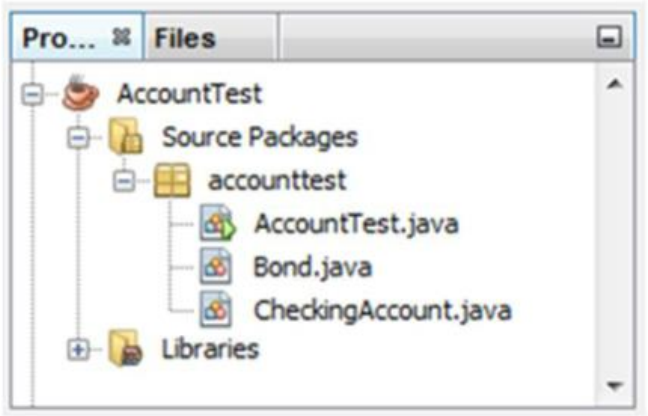
```
public ????? calcInterest(){
    double interest = balance*interestRate/12;
    return interest;
}
```

- ○ void
- ○ int
- ◉ double
- ○ String

Interactive Quizzes

Estructura del curso Java Foundations

Which statements are true, based on the following project directory?



- [ ] It's impossible to add more classes to this project.
- [x] Bond, and CheckingAccount are classes.
- [x] AccountTest.java, Bond.java, and CheckingAccount.java are files that exist in the same folder.
- [ ] An instance of a `CheckingAccount` cannot be instantiated.

Interactive Quizzes

Estructura del curso Java Foundations

Which two attempts at overloading the `printValues()` method cause a conflict?

```
public int printValues(String x, int y){
      ...
}
```

```
public void printValues(int i, String s){
      ...
}
```

```
public void printValues(int i){
      ...
}
```

```
public void printValues(String s, int i){
      ...
}
```

Interactive Quizzes

Estructura del curso Java Foundations

Drag and drop each box to correctly comment the following code.

```
public class Prisoner{

    private static int prisonerCount = 0;
    private int bookingNumber;


    public static void displayPrisonerCount(){
        System.out.println(prisonerCount);
    }
    public void callAnotherMethod(){
        displayPrisonerCount();
    }
}
```

//Static Variable

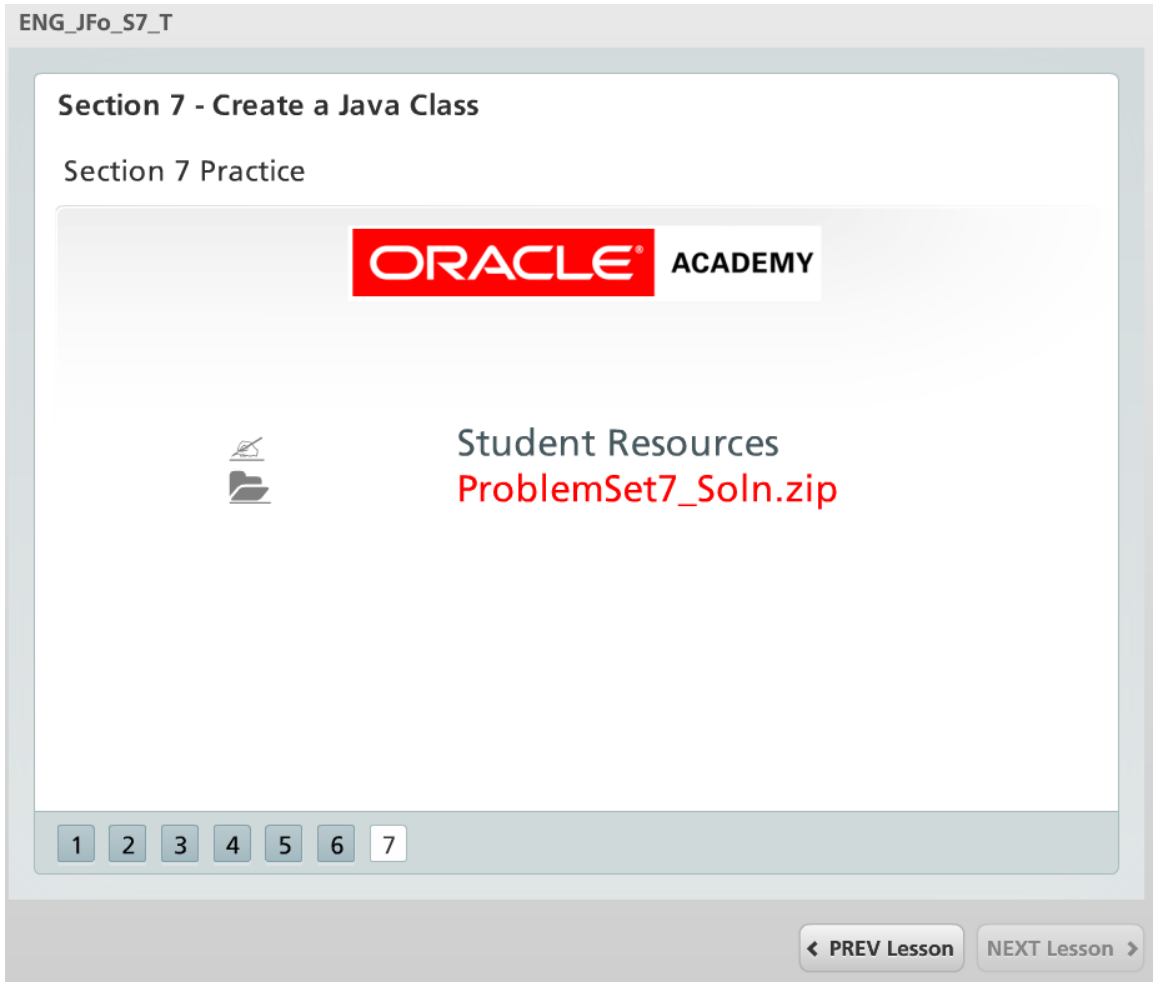
//Static Method

//Instance Variable

//Instance Method

Interactive Quizzes

Estructura del curso Java Foundations



ORACLE ACADEMY www.oracle.com/academy

Practices - Section 7: Date Night at the Arcade

Overview

Tonight is date night at the arcade. After great evening of playing games and winning prizes, you and your date can't help wondering "How are these machines programmed?". You discuss possible designs on the subway back to campus. You enjoy the rest of the night romantically programming your ideas together.

You've made several observations about the arcade. A terminal is used to convert money into game credits. Credits are loaded onto plastic game cards. This data is stored in a card's magnetic strip. Cards may be swiped at any arcade game through the game's magnetic card reader. Games subtract credits from a card, but awards tickets. Tickets are also stored on a card's magnetic strip. Tickets may be exchanged for prizes at the terminal. The terminal is also used to check a card's credit balance and ticket count, and to transfer credits or tickets between cards.

Tasks

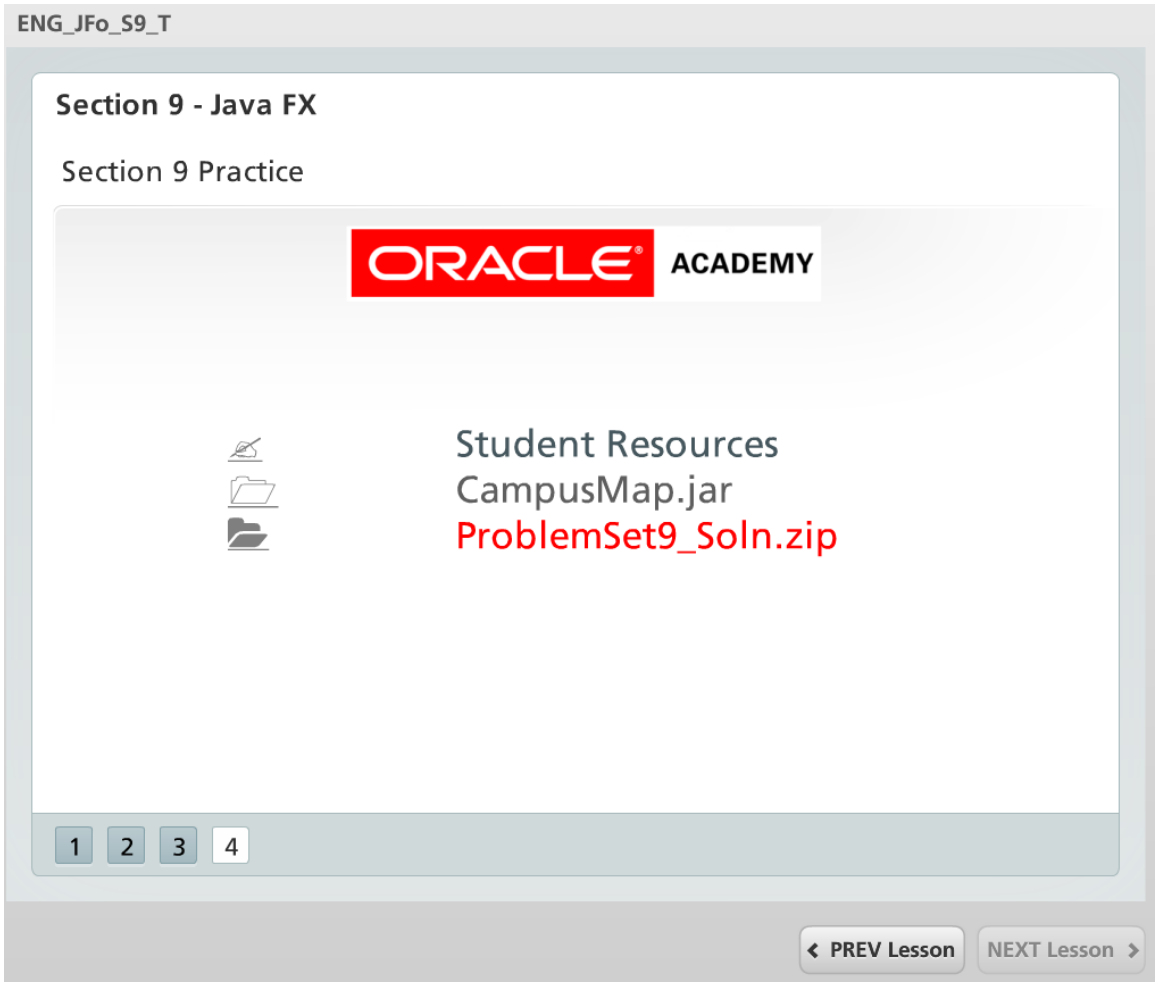
Write a Java program that models the properties, behaviors, and interactions of objects at the arcade. You'll also need a test class that contains a main method. Use the main method to model actions that would drive the program such as object instantiations and card swipes. All fields must be `private`. Provide getter and any necessary setter methods.

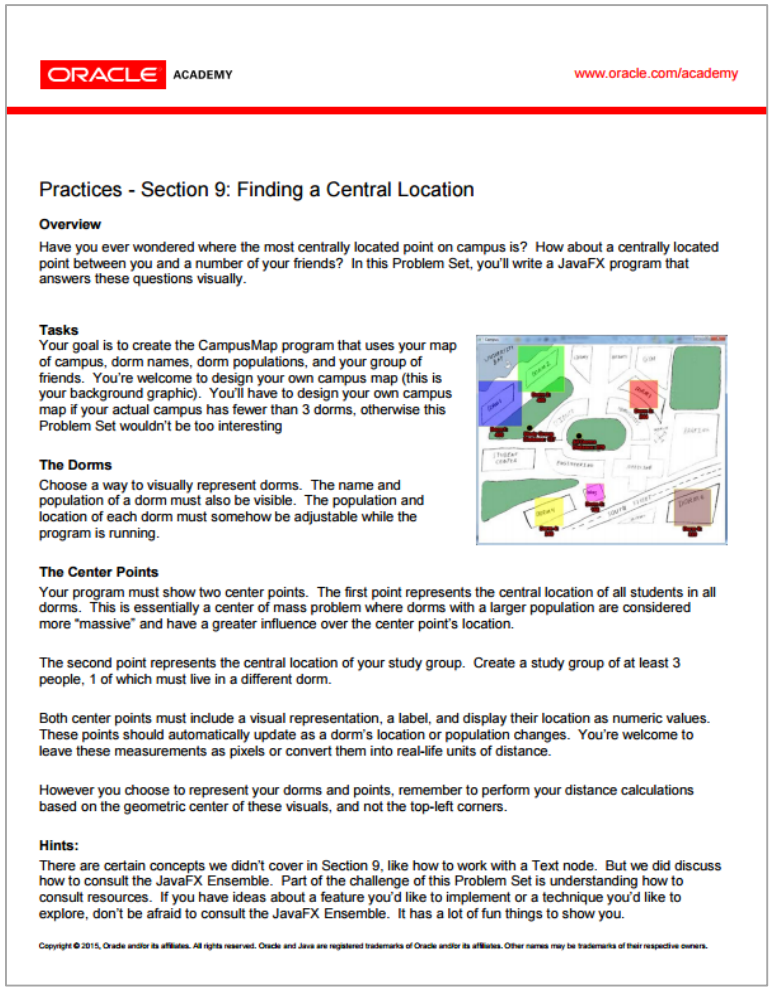
Cards

The magnetic strip on game cards offers limited storage space and zero computing power. Cards store information about their current credit balance, ticket balance, and card number. Neither balance should ever be negative. Individual cards are incapable of performing calculations, including simple addition, or realizing that their balances could go negative.

Práctica final de sección

Estructura del curso Java Foundations





ORACLE ACADEMY www.oracle.com/academy

Practices - Section 9: Finding a Central Location

Overview

Have you ever wondered where the most centrally located point on campus is? How about a centrally located point between you and a number of your friends? In this Problem Set, you'll write a JavaFX program that answers these questions visually.

Tasks

Your goal is to create the CampusMap program that uses your map of campus, dorm names, dorm populations, and your group of friends. You're welcome to design your own campus map (this is your background graphic). You'll have to design your own campus map if your actual campus has fewer than 3 dorms, otherwise this Problem Set wouldn't be too interesting

The Dorms

Choose a way to visually represent dorms. The name and population of a dorm must also be visible. The population and location of each dorm must somehow be adjustable while the program is running.

The Center Points

Your program must show two center points. The first point represents the central location of all students in all dorms. This is essentially a center of mass problem where dorms with a larger population are considered more "massive" and have a greater influence over the center point's location.

The second point represents the central location of your study group. Create a study group of at least 3 people, 1 of which must live in a different dorm.

Both center points must include a visual representation, a label, and display their location as numeric values. These points should automatically update as a dorm's location or population changes. You're welcome to leave these measurements as pixels or convert them into real-life units of distance.

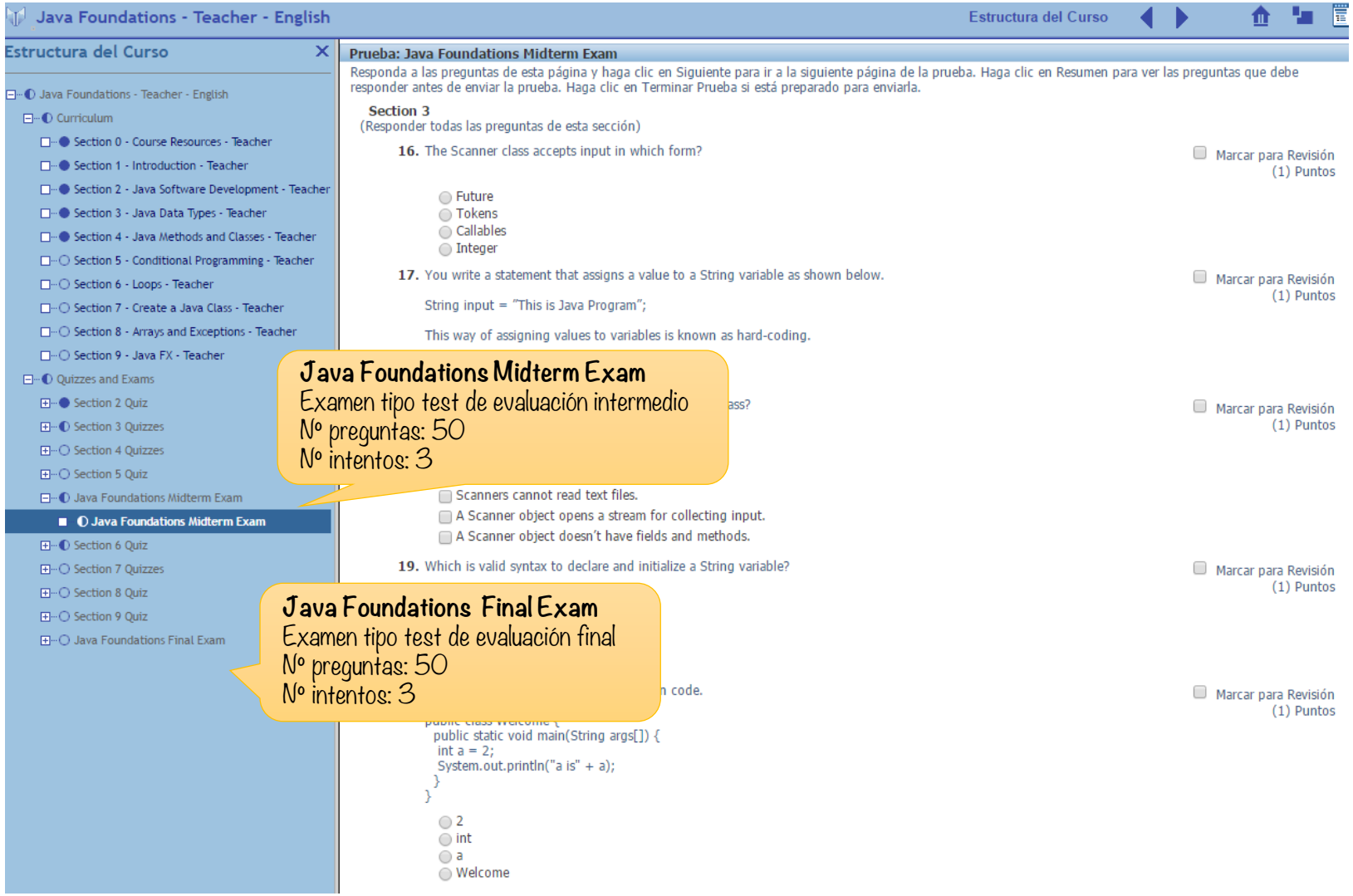
However you choose to represent your dorms and points, remember to perform your distance calculations based on the geometric center of these visuals, and not the top-left corners.

Hints:

There are certain concepts we didn't cover in Section 9, like how to work with a Text node. But we did discuss how to consult the JavaFX Ensemble. Part of the challenge of this Problem Set is understanding how to consult resources. If you have ideas about a feature you'd like to implement or a technique you'd like to explore, don't be afraid to consult the JavaFX Ensemble. It has a lot of fun things to show you.

Práctica final de sección

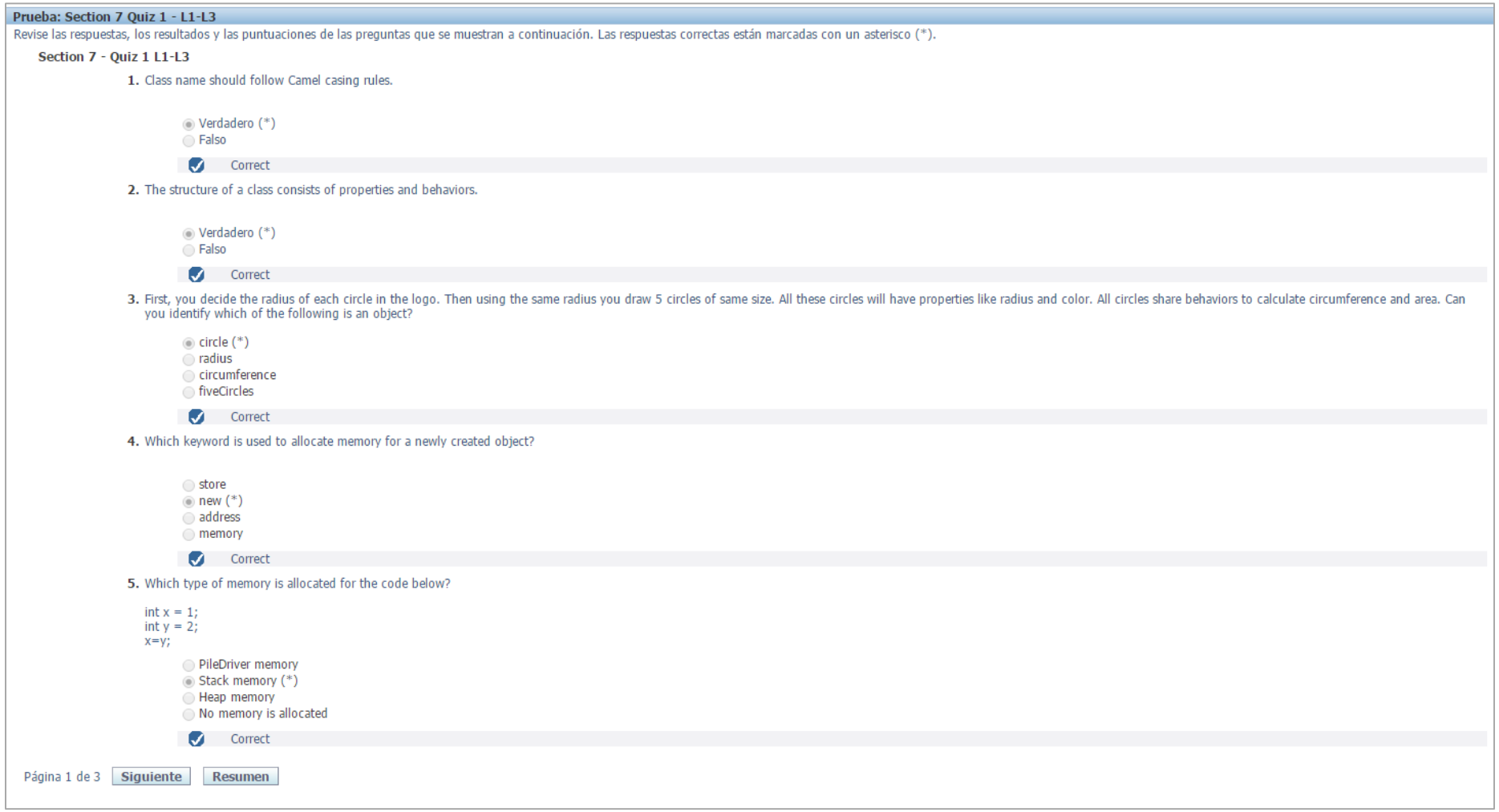
Estructura del curso Java Foundations



pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Estructura del curso Java Foundations



Prueba: Section 7 Quiz 1 - L1-L3

Revise las respuestas, los resultados y las puntuaciones de las preguntas que se muestran a continuación. Las respuestas correctas están marcadas con un asterisco (*).

Section 7 - Quiz 1 L1-L3

1. Class name should follow Camel casing rules.

- (•) Verdadero (*)
- ( ) Falso

Correct

2. The structure of a class consists of properties and behaviors.

- (•) Verdadero (*)
- ( ) Falso

Correct

3. First, you decide the radius of each circle in the logo. Then using the same radius you draw 5 circles of same size. All these circles will have properties like radius and color. All circles share behaviors to calculate circumference and area. Can you identify which of the following is an object?

- (•) circle (*)
- ( ) radius
- ( ) circumference
- ( ) fiveCircles

Correct

4. Which keyword is used to allocate memory for a newly created object?

- ( ) store
- (•) new (*)
- ( ) address
- ( ) memory

Correct

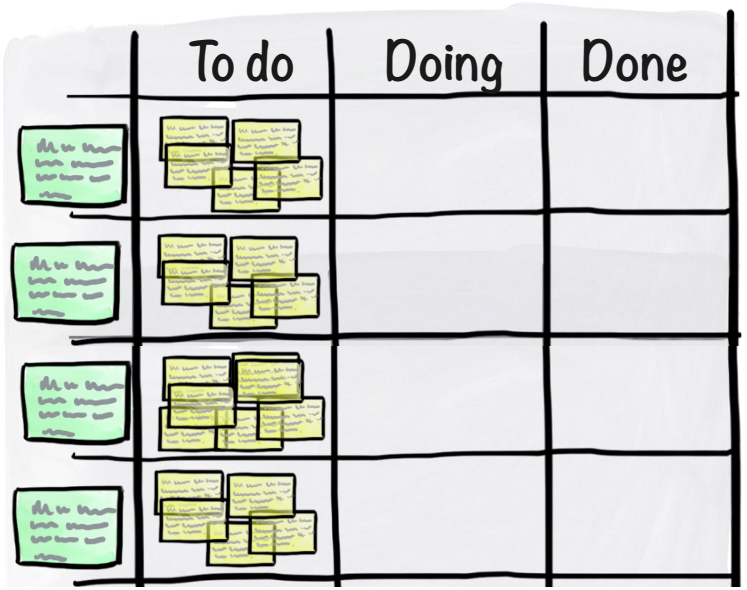
5. Which type of memory is allocated for the code below?

```
int x = 1;
int y = 2;
x=y;
```

- ( ) PileDriver memory
- (•) Stack memory (*)
- ( ) Heap memory
- ( ) No memory is allocated

Correct

Página 1 de 3 [Siguiente] [Resumen]



Kanban Board
(Visual Management)

Agenda

1.- Introducción
2.- Plan de formación y certificación en Java
3.- Curso Java Foundations
4.- Estructura del curso Java Foundations
5.- Demo: Curso Java Foundations
6.- Oracle iLearning: Informes y diplomas

Demo: Curso Java Foundations

Java Foundations

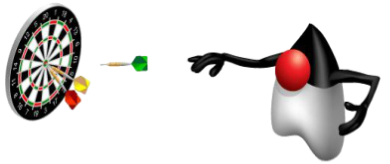
2-3
Introduction to Object-Oriented Programming Concepts

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

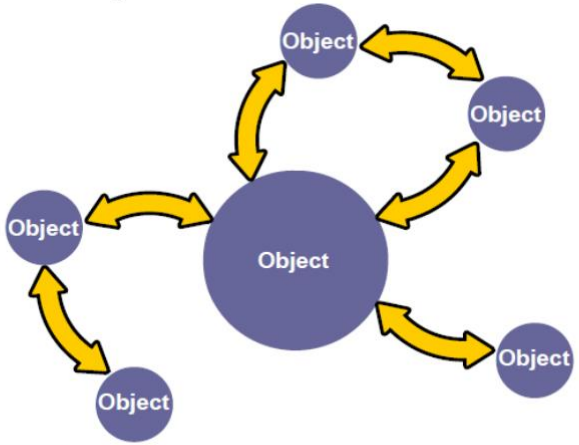
- Differentiate between procedural and object-oriented programming
- Understand a **class** as a blueprint for an **object**
- Understand a class is used to create **instances** of an object
- Model objects as a combination of …
  - **Properties** (data fields)
  - **Behaviors** (methods)



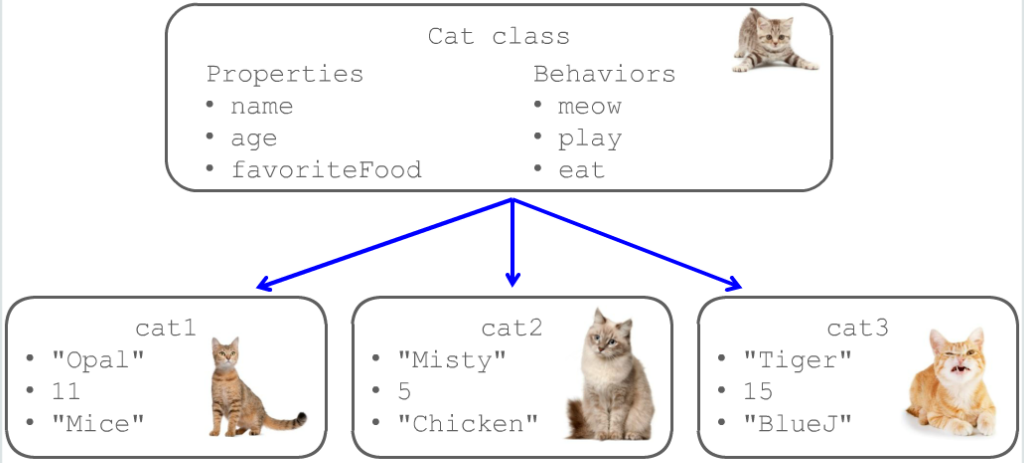
Demo: Curso Java Foundations

Object-Oriented Programming

- Interaction of objects
- No prescribed sequence

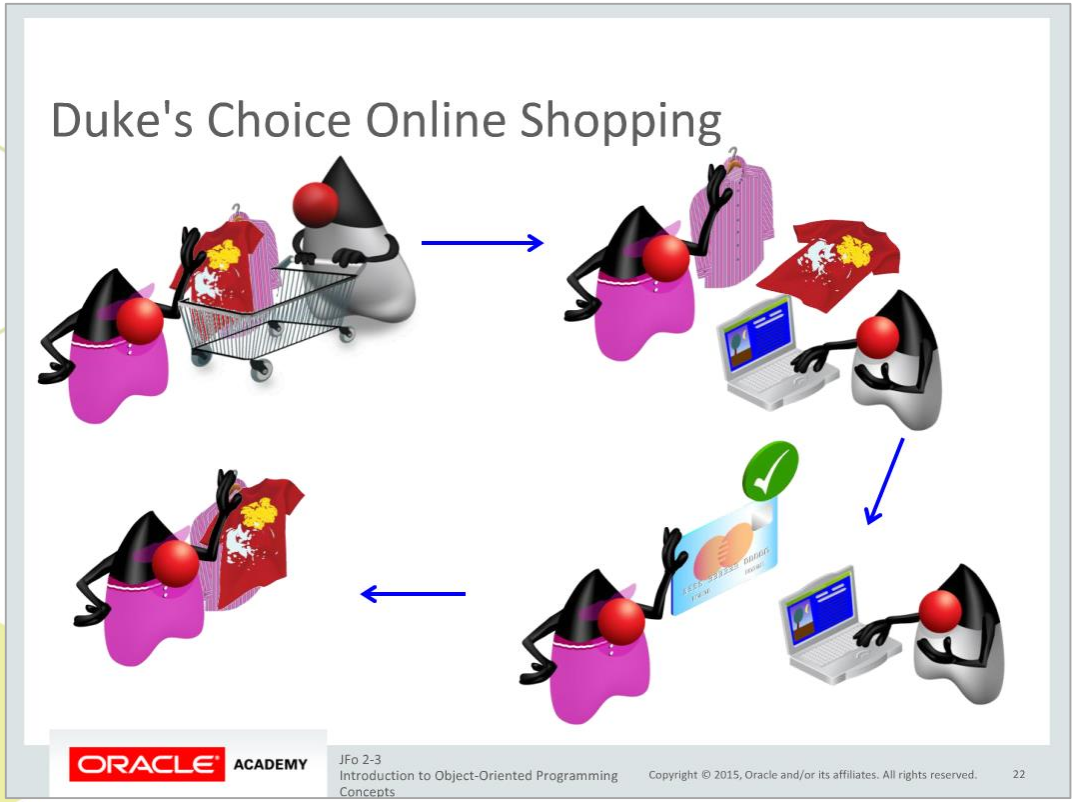


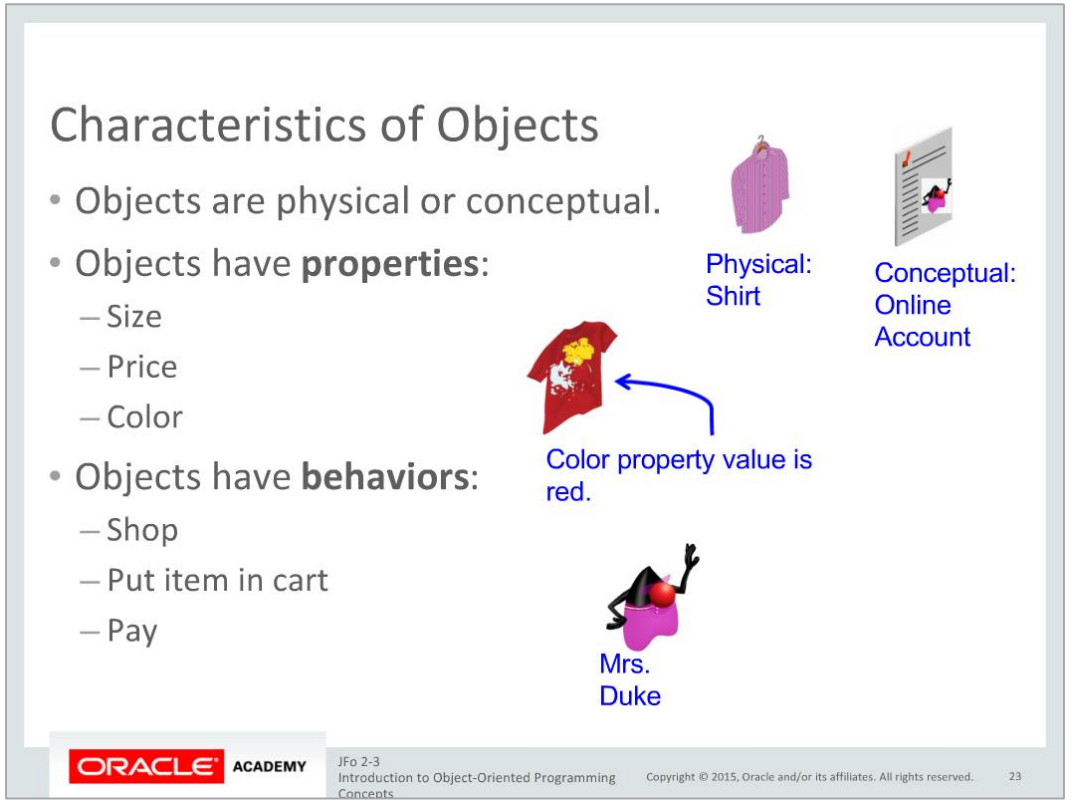
Creating New Instances from a Blueprint



All cat instances share the ability to meow, play, and eat.

Demo: Curso Java Foundations

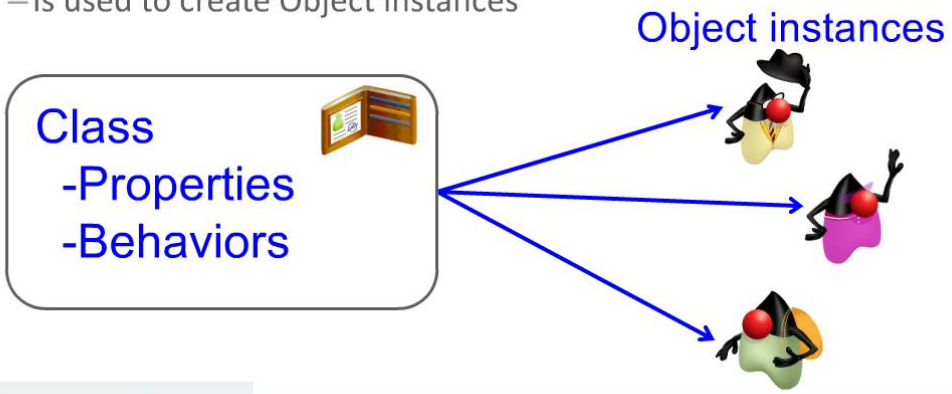




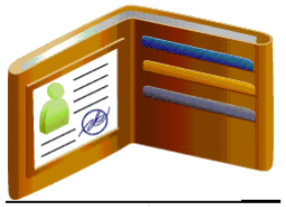
Demo: Curso Java Foundations

Classes and Instances

- Remember, a class …
  - Is a blueprint or recipe for an object
  - Describes an object's properties and behaviors
  - Is used to create Object instances



Customer Properties and Behaviors



- Properties:
  - Name
  - Address
  - Age
  - Order number
  - Customer number
- Behaviors:
  - Shop
  - Set address
  - Add item to cart
  - Ask for a discount
  - Display customer details

Demo: Curso Java Foundations

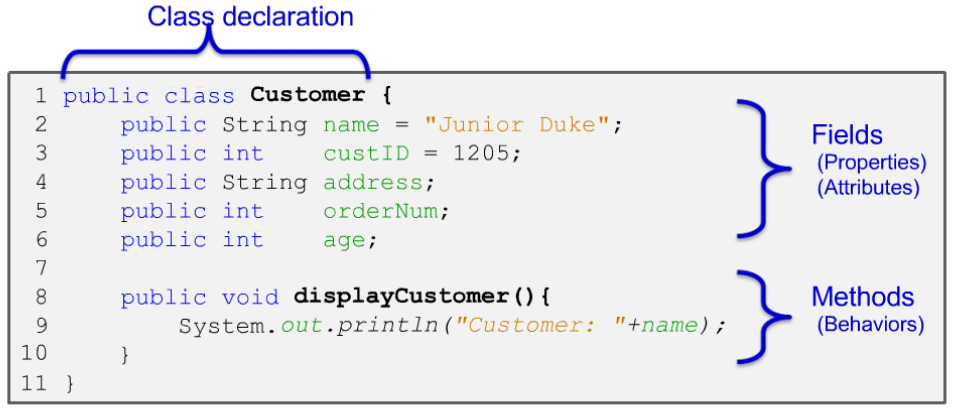
Translating into Java Syntax

Translating into Java Syntax

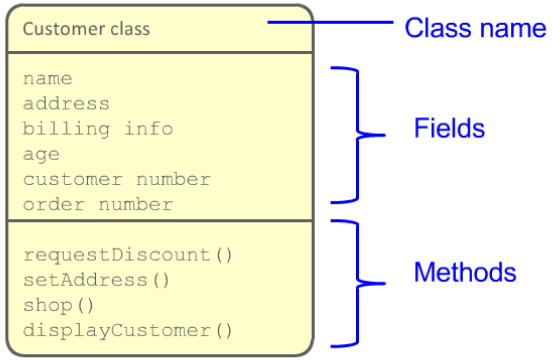
```
1  public class Customer {
2      public String name = "Junior Duke";
3      public int    custID = 1205;
4      public String address;
5      public int    orderNum;
6      public int    age;
7
8      public void displayCustomer(){
9          System.out.println("Customer: "+name);
10     }
11 }
```

Demo: Curso Java Foundations

Java Terminology



Modeling Properties and Behaviors



Demo: Curso Java Foundations

Exercise 2, Part 1

Given the following scenario, what objects could you potentially model to complete your program?

> Design a program for a coin-sorting machine. This machine should measure, count, and sort coins based on their size or value. It should also print a receipt.

- List at least 3 objects:

Exercise 2, Part 2

- Chose an object from Part 1.
- What properties and behaviors of this object could you include in your program?

- Properties:

- Behaviors:

Demo: Curso Java Foundations

Decide which bubbles represent properties, and which represent behaviors of a `Friend` Object. Drag the bubble into the corresponding box.

Drag and drop the term on the right to match the description on the left.

A blueprint for creating objects	Class
Can be physical or conceptual, and have properties and behaviors	Objects
Characteristics that can describe an object	Properties
Actions and interactions an object is capable of	Behaviors
In individual object created from a blueprint	Instance

Demo: Curso Java Foundations

Java Foundations

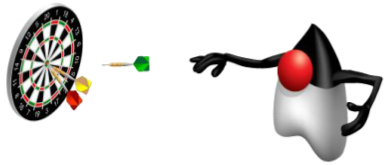
7-1
Creating a Class

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- Create a Java test/main class
- Create a Java class in NetBeans
- Use conditionals in methods
- Translate specifications or a description into fields and behaviors

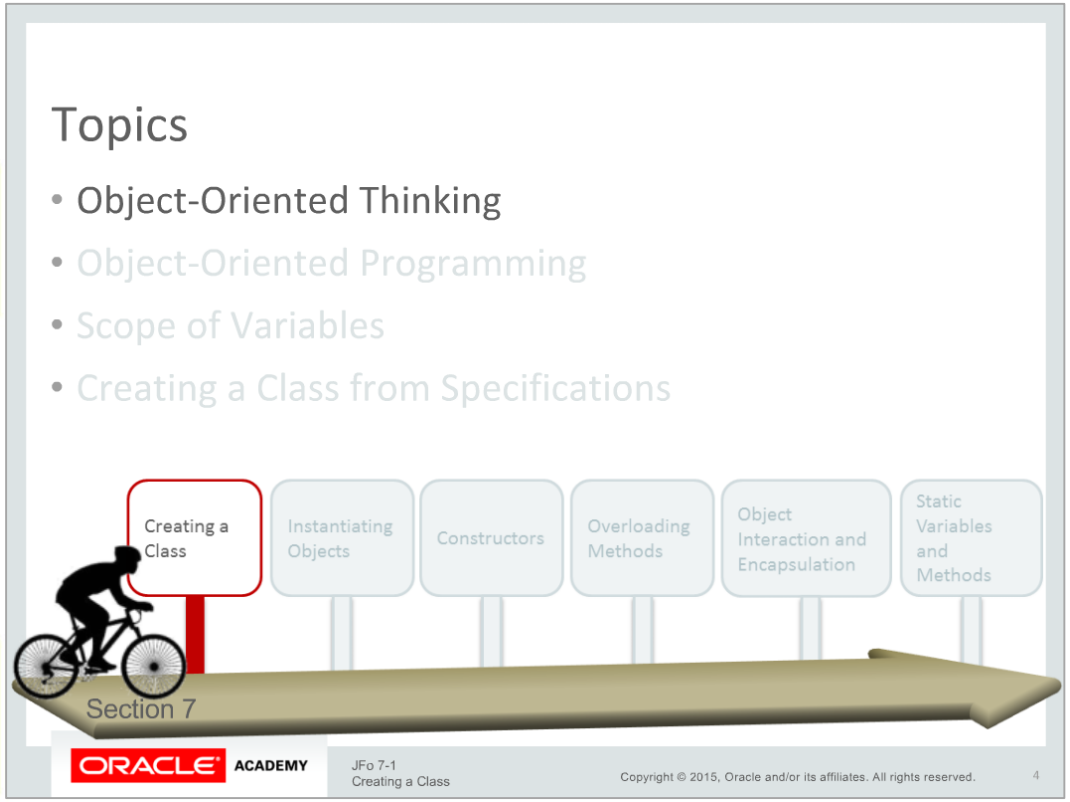


ORACLE ACADEMY

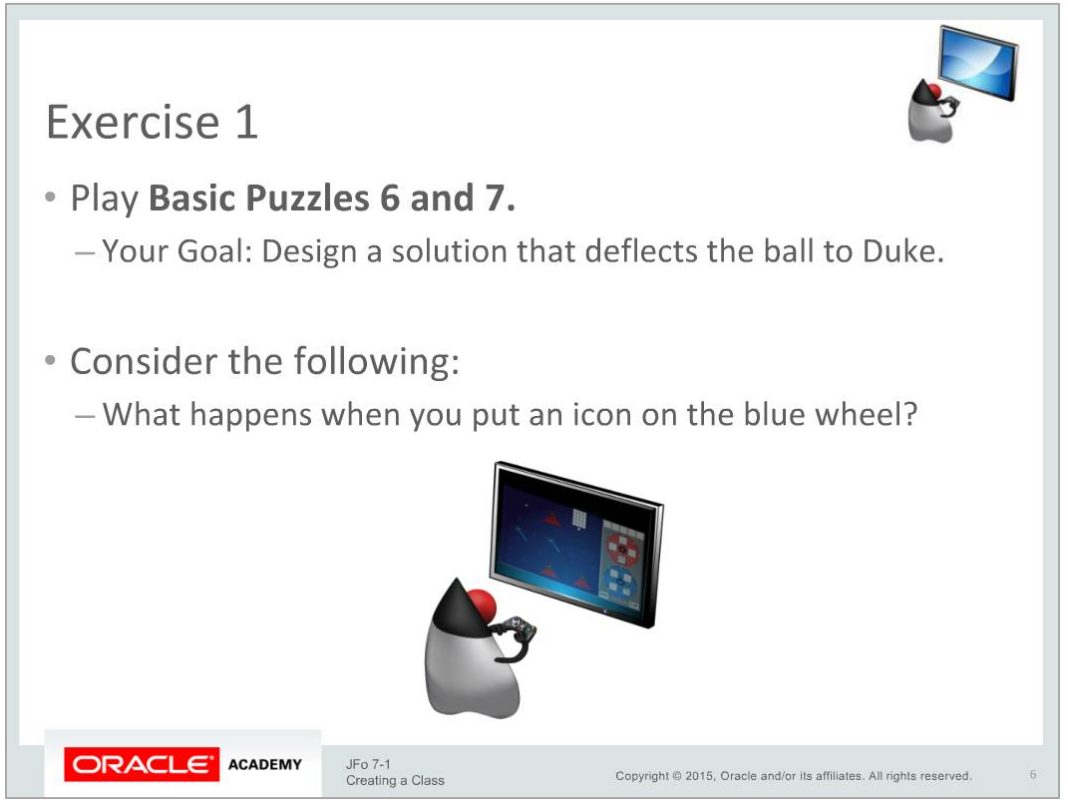
pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Demo: Curso Java Foundations

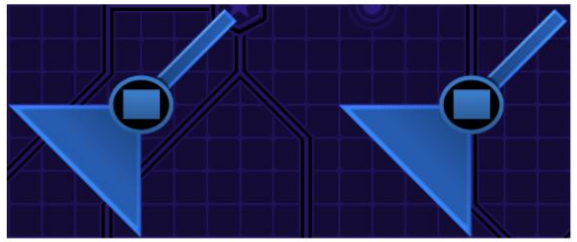


Demo: Curso Java Foundations



Demo: Curso Java Foundations

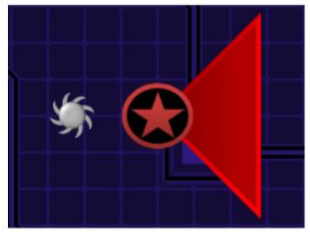
Describing a Blue Bumper



(Fields)
- Properties:
  - Color
  - Shape
  - x-position
  - x-position

(Methods)
- Behaviors:
  - Make ping sound
  - Flash
  - Deflect ball
  - Get destroyed

What If the Ball Collides with a Bumper?



A method with the following logic is called:

```
public void onCollisionWithBall(Ball ball){
    if(ball.isBlade == true){          //Ball is blade
        getDestroyed();
    }
    else{                              //Ball is not blade
        deflectBall();
    }
}
```

Demo: Curso Java Foundations

Modeling a Savings Account

- You could model one savings account like this:

```
public class SavingsAccount{

   public static void main(String args[]){

        int balance = 1000;
        String name = "Damien";
   }
}
```

- And two accounts like this:

```
        int balance1 = 1000;
        String name1 = "Damien";

        int balance2 = 2000;
        String name2 = "Bill";          //Copy, Paste, Rename
```


How to Structure a Class

- Code should fit this format:

```
1 public class SavingsAccount {
2
3
4
5
6
7
8
9 }
```

Properties

Behaviors

Demo: Curso Java Foundations

How to Structure a Class

- Code should fit this format:

```
1 public class SavingsAccount {
2     public double balance;
3     public double interestRate = 0.01;
4     public String name;
5
6     public void displayCustomer(){
7         System.out.println("Customer: "+name);
8     }
9 }
```

- With one simple line of code(line 3), all 1000 accounts have an interest rate.
  - And we can change the rate at any time for any account.

The Main Method as a Driver Class

- Place the main method in a test class.
  - The main method is often used for instantiation.

```
1 public class AccountTest {
2     public static void main(String[] args){
3
4         SavingsAccount sa0001 = new SavingsAccount();
5         sa0001.balance = 1000;
6         sa0001.name = "Damien";
7         sa0001.interestRate = 0.02;
8
9         SavingsAccount sa0002 = new SavingsAccount();
10        sa0001.balance = 2000;
11        sa0001.name = "Bill";
12 }
```

Demo: Curso Java Foundations

Variable Scope

- Fields are accessible anywhere in a class.
  - This includes within methods.

```
public class SavingsAccount {
    public double balance;
    public double interestRate;
    public String name;

    public void displayCustomer(){
        System.out.println("Customer: "+name);
        System.out.println("Balance: " +balance);
        System.out.println("Rate: "    +rate);
    }
}
```


Exercise 2

- Create a new Java project.
- Create an `AccountTest` class with a main method.
- Create a `CheckingAccount` class.
  - Include fields for `balance` and `name`.
- Instantiate a `CheckingAccount` object from the main method.
  - Assign values for this object's `balance` and `name` fields.

Demo: Curso Java Foundations

Variable Scope

- Variables created within a method cannot be accessed outside that method.
  - This includes methods parameters.

```
public class SavingsAccount {
    public double balance;
    public double interestRate;
    public String name;

    public void deposit(int x){
        balance += x;
    }

    public void badMethod(){
        System.out.println(x);
    }
}
```

Scope of x

Not scope of x

Accessing Fields and Methods from Another Class

1. Create an instance.
2. Use the dot operator (.)

```
public class AccountTest {
    public static void main(String[] args){

        SavingsAccount sa0001 = new SavingsAccount();
        sa0001.name = "Damien";
        sa0001.deposit(1000);
}
```

1)

2)

```
public class SavingsAccount {
    public String name;
    public double balance;

    public void deposit(int x){
        balance += x;
    }
}
```

Demo: Curso Java Foundations

Passing Values to Methods

- 1000 is passed to the `deposit()` method.
- The value of `x` becomes 1000.

```
public class AccountTest {
     public static void main(String[] args){

          SavingsAccount sa0001 = new SavingsAccount();
          sa0001.name = "Damien";
          sa0001.deposit(1000);
}
```

```
public class SavingsAccount {
     public String name;
     public double balance;

     public void deposit(int x){
         balance += x;
     }
}
```

x = 1000

Exercise 3

- Continue editing the `AccountTest` project.
- Write a `withdraw()` method for checking accounts that …
  - Accepts a double argument for the amount to be withdrawn.
  - Prints a warning if the balance is too low to make the withdrawal.
  - Prints a warning if the withdrawal argument is negative.
  - If there are no warnings, the withdrawal amount is subtracted from the balance. Print the new balance.
- Test this method with the instance from Exercise 2.

Demo: Curso Java Foundations

What if I Need a Value from a Method?

- Variables are restricted by their scope.
- But it's still possible to get the **value** of these variables out of a method.

```
public class SavingsAccount {
    public double balance;
    public double interestRate;
    public String name;

    public void calcInterest(){
        double interest = balance*interestRate/12;

    }
}
```

Scope of interest

Returning Values **from** Methods

- If you want to get a value from a method …
  - Write a return statement.
  - Change the method type from `void` to the type that you want returned.

```
public class SavingsAccount {
    public double balance;
    public double interestRate;
    public String name;

    //This method has a double return type
    public double calcInterest(){
        double interest = balance*interestRate/12;
        return interest;
    }
}
```

Demo: Curso Java Foundations

Returning Values: Example

- When `getInterest()` returns a value …

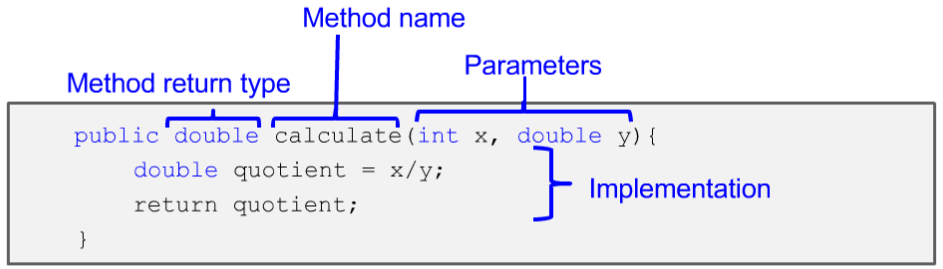
```
public class AccountTest {
     public static void main(String[] args){
          SavingsAccount sa0001 = new SavingsAccount();
          sa0001.balance = 1000;
          sa0001.balance += sa0001.calcInterest();
}
```

- It's the equivalent of writing …

```
public class AccountTest {
     public static void main(String[] args){
          SavingsAccount sa0001 = new SavingsAccount();
          sa0001.balance = 1000;
          sa0001.balance += 0.83;
}
```

  - But it's better and more flexible because the value is calculated instead of hard-coded.

Summary About Methods



Demo: Curso Java Foundations

Limiting the Main Method

- The main method should be as small as possible.
- The example below isn't very good because …
  - Increasing an account's balance based on interest is a typical behavior of accounts.
  - The code for this behavior should instead be written as a method within the `SavingsAccount` class.
  - It's also dangerous to have an account program where the balance field can be freely manipulated.

```
public class AccountTest {
     public static void main(String[] args){
          SavingsAccount sa0001 = new SavingsAccount();
          sa0001.balance = 1000;
          sa0001.balance += sa0001.calcInterest();
}
```


Exercise 4

- Continue editing the `AccountTest` project.
- Create a new class according to the description. Be sure to instantiate this class and test its methods.

Create a Savings Bond. A person may purchase a bond for any term between 1 and 60 months. A bond earns interest every month until its term matures (0 months remaining). The term and interest rate are set at the same time. The bond's interest rate is based on its term according to the following tier system:

- 0–11 months: 0.5%
- 12–23 months: 1.0%
- 24–35 months: 1.5%
- 36–47 months: 2.0%
- 48–60 months: 2.5%

Demo: Curso Java Foundations

Describing a Savings Bond

- Properties:
  - Name
  - Balance
  - Term
  - Months Remaining
  - Interest Rate
- Behaviors:
  - Set Interest Rate Based on Term
  - Earn Interest
  - Mature (0 months remaining)

Translating to Java Code: Part 1

- Your `Bond` class may have represented fields like this:

```
public class Bond{
    public String name;
    public double balance, rate;
    public int term, monthsRemaining;

…
```

Demo: Curso Java Foundations

Translating to Java Code: Part 2

- And include the following methods:

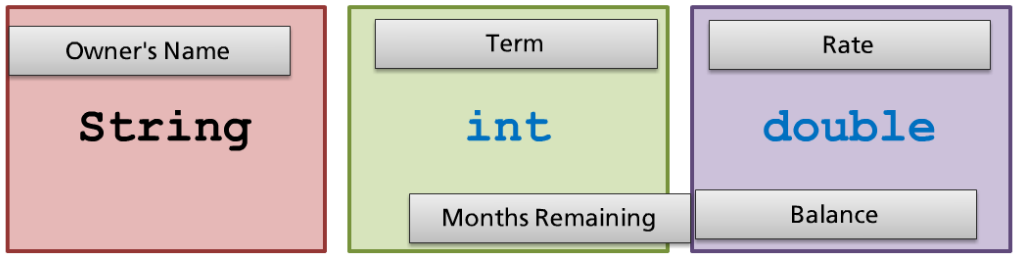
```
public void setTermAndRate(int t){
    if(t>=0 && t<12)
        rate = 0.005;
    else if(t>=12 && t<24)
        rate = 0.010;
    else if(t>=24 && t<36)
        rate = 0.015;
    else if(t>=36 && t<48)
        rate = 0.020;
    else if(t>=48 && t<=60)
        rate = 0.025;
    else{
        System.out.println("Invalid Term");
        t = 0;
    }
    term = t;
    monthsRemaining = t;
}
```


Translating to Java Code: Part 3

```
…
    public void earnInterest(){
        if(monthsRemaining > 0){
            balance += balance*rate/12;
            monthsRemaining--;
            System.out.println("Balance: $" +balance);
            System.out.println("Rate: " +rate);
            System.out.println("Months Remaining:"
                    +monthsRemaining);
        }
        else{
            System.out.println("Bond Matured");
        }
    }
}
```

Demo: Curso Java Foundations

Savings Bonds have several properties. How could these properties be translated and represented by Java code? Match each property with the best data type.



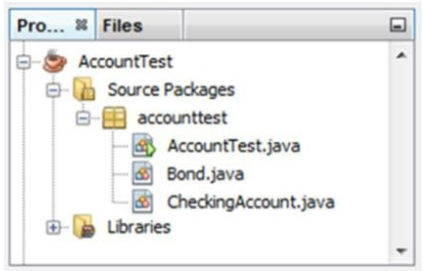
What should be the return type for this method, which returns a calculated value?

```
public ????? calcInterest(){
    double interest = balance*interestRate/12;
    return interest;
}
```

- ○ void
- ○ int
- ◉ double
- ○ String

Demo: Curso Java Foundations

Which statements are true, based on the following project directory?

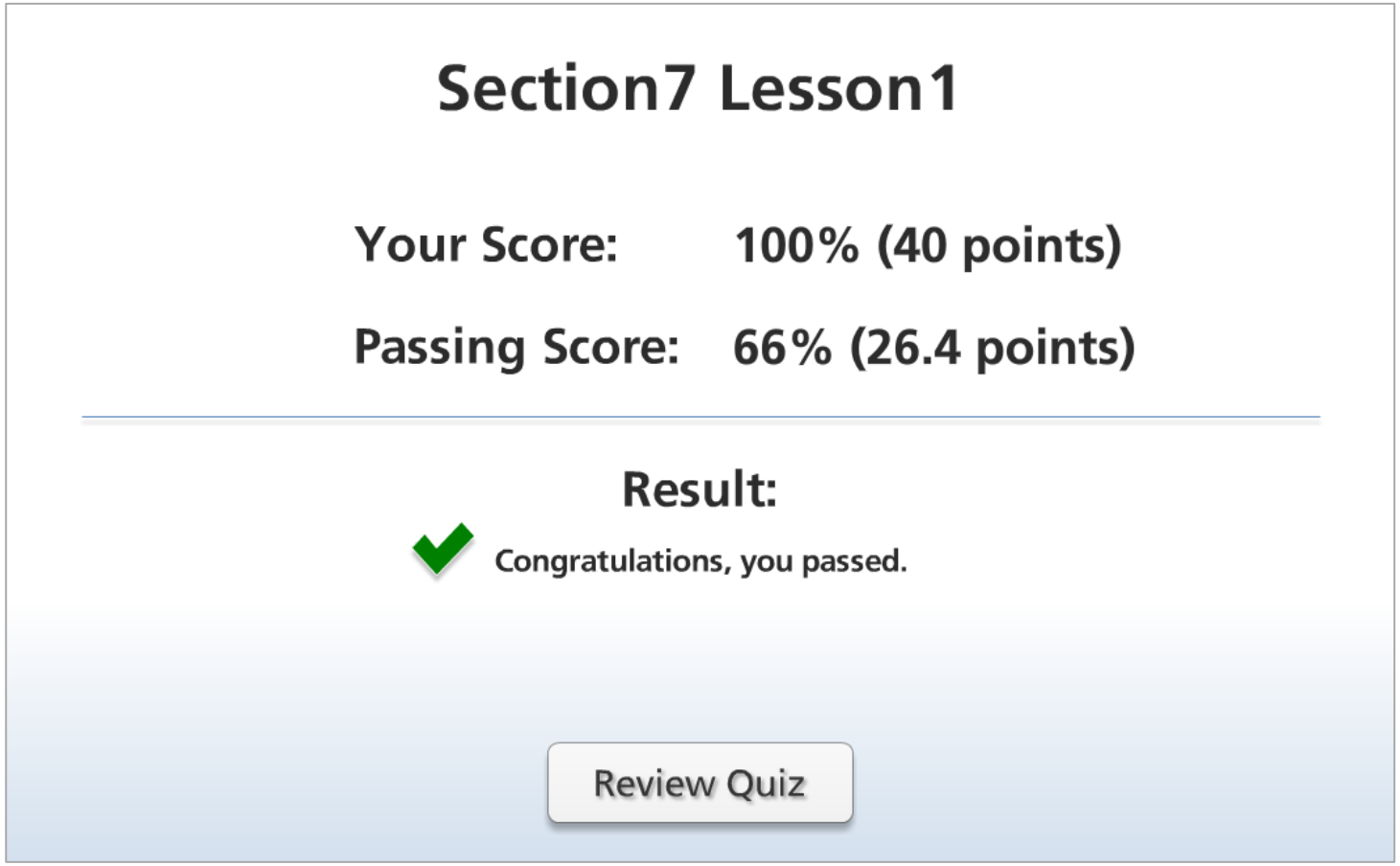


- [ ] It's impossible to add more classes to this project.
- [x] Bond, and CheckingAccount are classes.
- [x] AccountTest.java, Bond.java, and CheckingAccount.java are files that exist in the same folder.
- [ ] An instance of a `CheckingAccount` cannot be instantiated.

Which statements are true of the main method?

- [ ] The main method is stupid. Programs don't need a main method.
- [x] The main method should be as short as possible.
- [x] It's common to place the main method in a test class.
- [ ] The main method is one of many behaviors that an object may have.

Demo: Curso Java Foundations



Demo: Curso Java Foundations

Java Foundations

7-2
Instantiating Objects

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

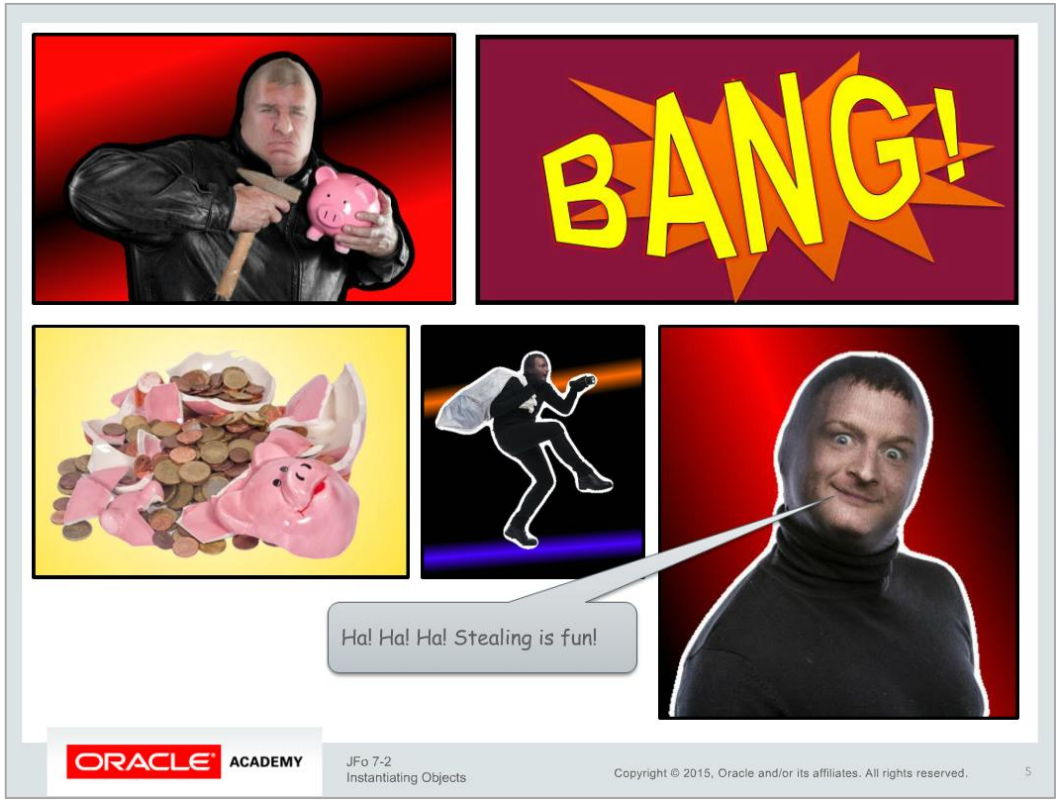
- Understand the memory consequences of instantiating objects
- Understand object references
- Understand the difference between stack and heap memory
- Understand how `Strings` are special objects

ORACLE ACADEMY

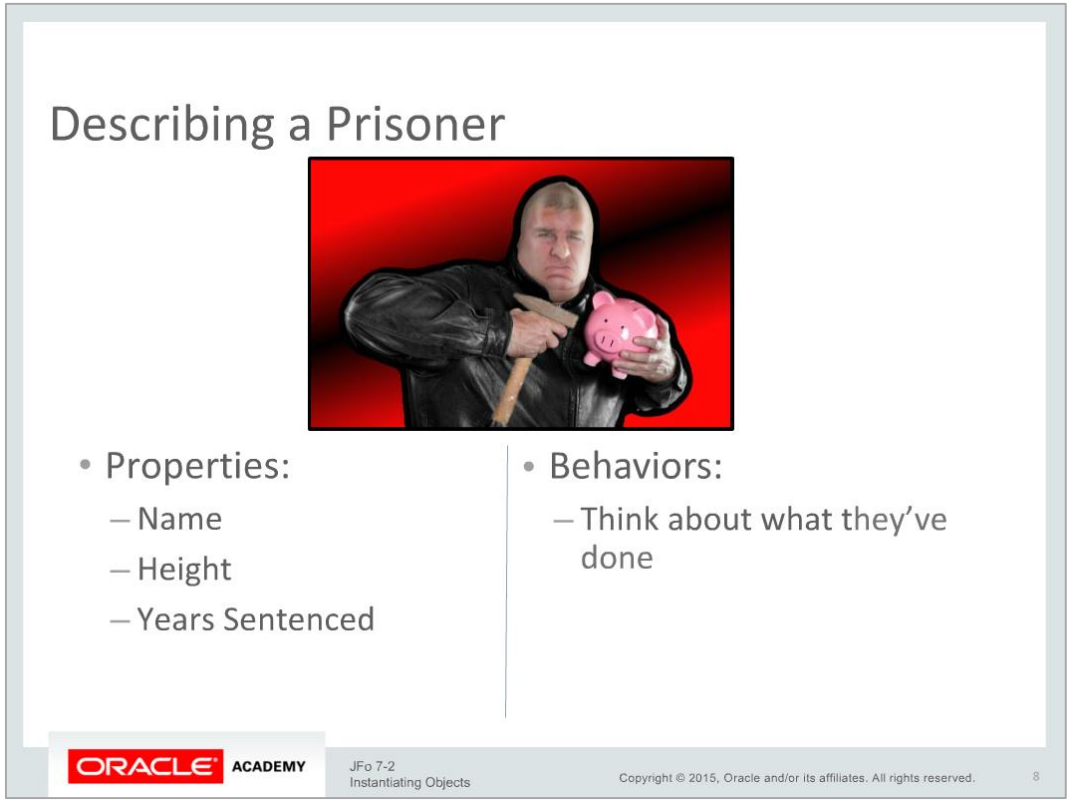
pue IMPULSANDO EL CONOCIMIENTO TIC CUALIFICADO

#PUEDAY16

Demo: Curso Java Foundations



Demo: Curso Java Foundations



Demo: Curso Java Foundations

Exercise 1, Part 1

- Create a new Java project.
- Create a `PrisonTest` class with a main method.
- Create a `Prisoner` class based on the description in the previous slide.
- Instantiate two prisoners and assign them the following properties:

```
Variable:  bubba
Name:      Bubba
Height:    6'10"   (2.08m)
Sentence:  4 years
```

```
Variable:  twitch
Name:      Twitch
Height:    5'8"    (1.73m)
Sentence:  3 years
```


Exercise 1, Part 2

Can prisoners fool security by impersonating each other?

- Write a print statement with a boolean expression that tests if `bubba == twitch`.
- Change the properties of `twitch` so that they match `bubba`.
- Then test the equality of these objects again.

```
Variable:  bubba
Name:      Bubba
Height:    6'10"   (2.08m)
Sentence:  4 years
```

```
Variable:  twitch
Name:      Bubba
Height:    6'10"   (2.08m)
Sentence:  4 years
```

Demo: Curso Java Foundations

Programming the Prisoner Class

Your class may look something like this:

```
public class Prisoner {
    public String name;
    public double height;
    public int sentence;

    public void think(){
        System.out.println("I'll have my revenge.");
    }
}
```


Prisoner Impersonation

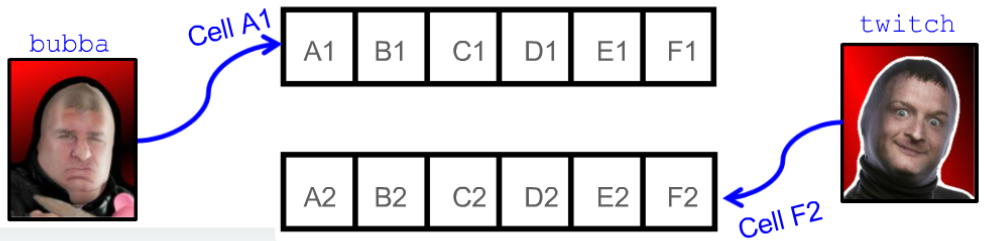
- The boolean `bubba == twitch` is `false`.
  - Security wasn't fooled by prisoners who share the same properties.
  - Security understood that each prisoner was a unique object.
- How is this possible?

```
public class PrisonTest {
    public static void main(String[] args){
        Prisoner bubba = new Prisoner();
        Prisoner twitch = new Prisoner();
        …
        System.out.println(bubba == twitch);      //false
    }
}
```

Demo: Curso Java Foundations

Prisoner Locations

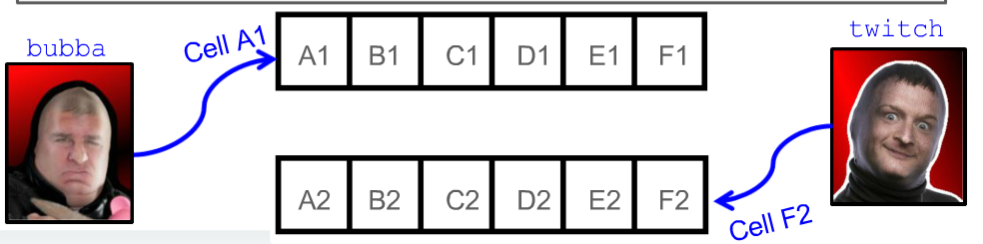
- Prisoners live in cells.
- New prisoners are assigned an available cell for living quarters.
- If a prisoner lives in a unique cell, he's a unique object.



Prisoner Object Locations

- Cells are like locations in memory.
- Instantiating a `Prisoner` fills an available location in memory with the new `Prisoner` object.

```
public class PrisonTest {
    public static void main(String[] args){
        Prisoner bubba = new Prisoner();
        Prisoner twitch = new Prisoner();
    }
}
```



Demo: Curso Java Foundations

The `new` Keyword

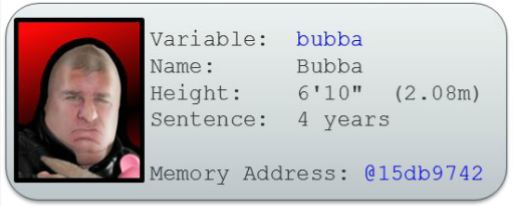
- The `new` keyword allocates available memory to store a newly created object.
- Java developers don't need to know an object's location in memory.
  - We only need to know the variable for the object.
  - But we can still print memory addresses.

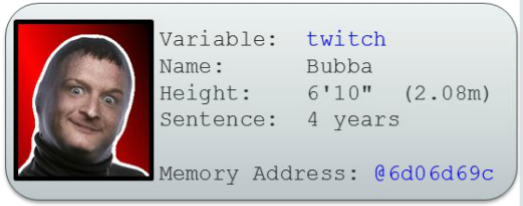
```
public class PrisonTest {
    public static void main(String[] args){
        Prisoner bubba = new Prisoner();
        Prisoner twitch = new Prisoner();
        System.out.println(bubba);      //prisontest.Prisoner@15db9742
        System.out.println(twitch);     //prisontest.Prisoner@6d06d69c
    }
}
```

Memory addresses

Objects with the Same Properties

- Objects may share the same properties.
- But it doesn't mean that these objects are equal.
- As long as you use the `new` keyword during instantiation …
  - You'll have unique objects.
  - Each object will have a different location in memory.





Demo: Curso Java Foundations

Comparing Objects

- If you compare two objects using the == operator …
  - You're checking if their **memory addresses** are equal.
  - You're **not** checking if their fields are equal.
- The boolean `bubba == twitch` is `false` because …
  - Memory addresses `@15db9742` and `@6d06d69c` are different.
  - It doesn't matter if `bubba` and `twitch` share the same properties.

```
public class PrisonTest {
    public static void main(String[] args){
        Prisoner bubba = new Prisoner();
        Prisoner twitch = new Prisoner();
        …
        System.out.println(bubba == twitch);      //false
    }
}
```


Accessing Objects by Using a Reference

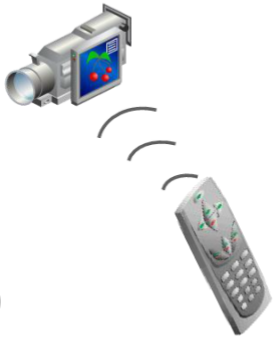
The camera is like the **object** that's accessed by using a reference.

The remote is like the **reference** that's used to access the camera.

Demo: Curso Java Foundations

Working with Object References

1. Pick up remote to gain access to the camera.



2. Press remote controls to have the camera do something.

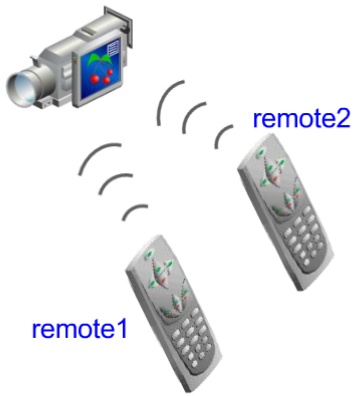
1. Create a Camera object and get a reference to it.

```
Camera remote1 = new Camera();
```

2. Call a method to have the Camera object do something.

```
remote1.play();
```


Working with Object References: Example 2



There's only one Camera object.

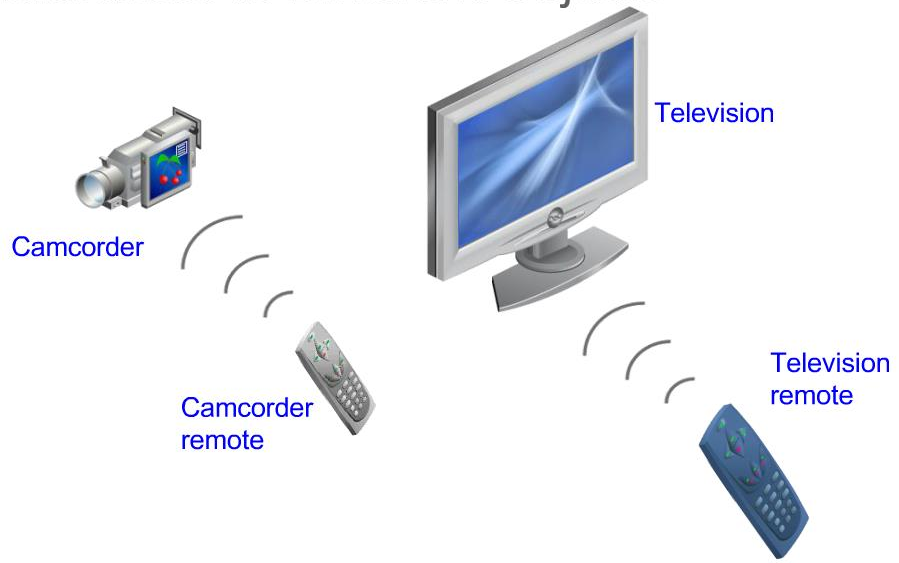
```
Camera remote1 = new Camera();

Camera remote2 = remote1;

remote1.play();
remote2.stop();
```

Demo: Curso Java Foundations

References to Different Objects



References to Different Objects: Example

Reference variable
Reference type
Object type

```
Camera remote1 = new Camera();
remote1.menu();

TV remote2 = new TV();
remote2.menu();

Prisoner bubba = new Prisoner();
bubba.think();
```

Demo: Curso Java Foundations

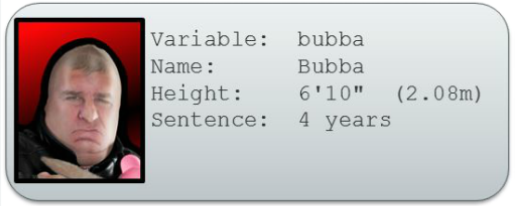
References to Different Objects: Example

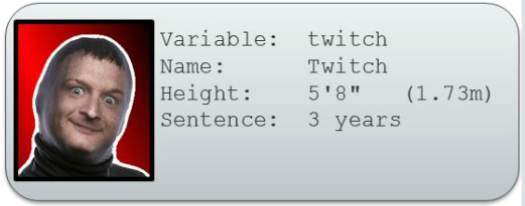
- The following example isn't allowed because …
  - The **Reference Type** doesn't match the **Object Type.**
  - A prisoner and a TV are completely different things.

```
Prisoner twitch = new TV();
```


Exercise 2

- Continue experimenting with the `PrisonTest` class.
- Is security fooled when reference variables change?
  - Instantiate two prisoners and assign them the properties below.
  - Test the equality of these objects.
  - Then set the reference variable for `bubba` equal to `twitch`.
  - Test the equality of these objects again.





Demo: Curso Java Foundations

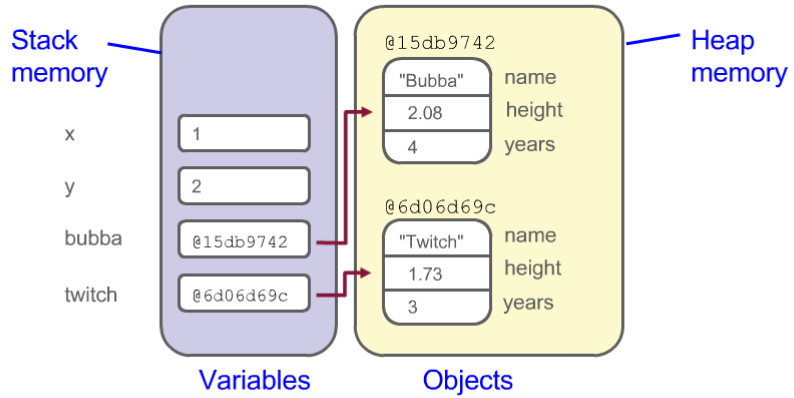
Stack Memory and Heap Memory

Understanding the results of Exercise 2 requires an understanding of the types of memory that Java uses.

- **Stack memory** is used to store …
  - Local variables
  - Primitives
  - References to locations in the heap memory
- **Heap memory** is used to store …
  - Objects

References and Objects in Memory

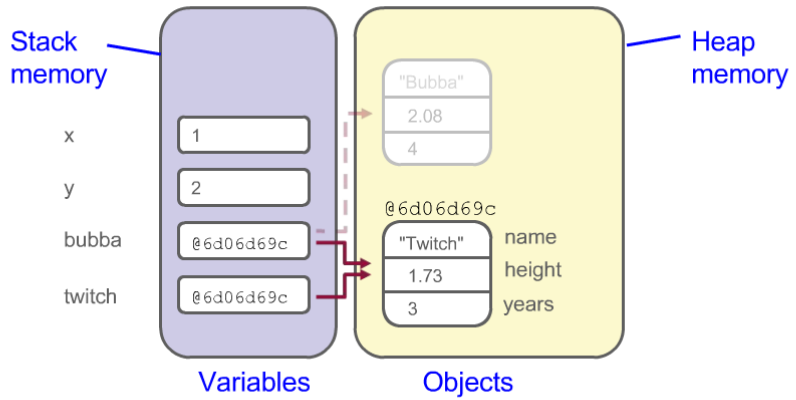
```
int x = 1;
int y = 2;
Prisoner bubba = new Prisoner();
Prisoner twitch = new Prisoner();
…
```



Demo: Curso Java Foundations

Assigning a Reference to Another Reference

```
bubba = twitch;
```



Two References, One Object

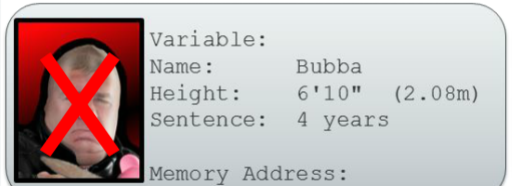
- As of line 14, `bubba` and `twitch` reference the same object.
- Either reference variable could be used to access the same data.

```
11 Prisoner bubba = new Prisoner();
12 Prisoner twitch = new Prisoner();
13
14 bubba = twitch;
15
16 bubba.name = "Bubba";
17 twitch.name = "Twitch";
19
20 System.out.println(bubba.name);       //Twitch
21 System.out.println(bubba == twitch); //true
```

Demo: Curso Java Foundations

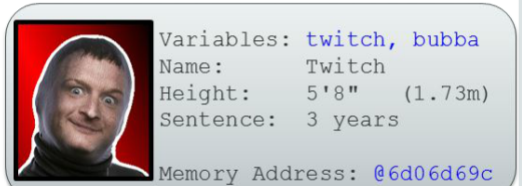
What Happened to Bubba?

- If no more reference variables point to an object …
- Java **automatically** clears the memory once occupied by that object.
  - This is called **Garbage Collection.**
  - The data associated with this object is lost forever.



```
Variable:
Name:       Bubba
Height:     6'10"   (2.08m)
Sentence:   4 years

Memory Address:
```



```
Variables: twitch, bubba
Name:       Twitch
Height:     5'8"    (1.73m)
Sentence:   3 years

Memory Address: @6d06d69c
```


Strings Are Special Objects

- Printing a `String` reference prints the actual `String` instead of the object's memory address.
- `String`s can be instantiated with the `new` keyword.
  - But you shouldn't do this.

```
String s1 = new String("Test");
```

- `String`s should be instantiated without `new`.
  - This is more memory-efficient.
  - We'll explore why in the next few slides.

```
String s2 = "Test";
```

Demo: Curso Java Foundations

Exercise 3

- Continue experimenting with the `PrisonTest` class.
- See the memory consequences of `String`s for yourself.
  - Instantiate two prisoners with the names shown below.
  - Set their names by using the `new` keyword and test the equality of these `String`s by using ==.
  - Set their names without using the `new` keyword and test the equality of these `String`s by using ==.

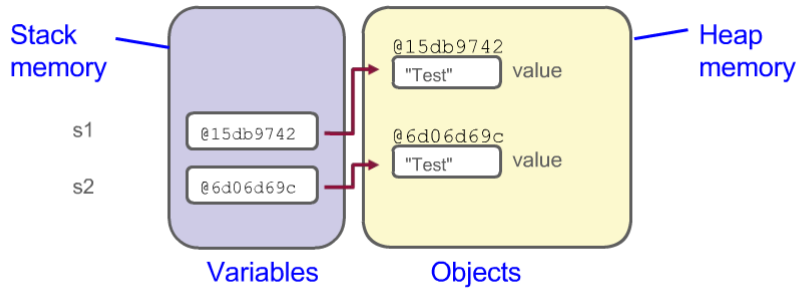
```
Variable:  bubba
Name:      Bubba
Height:    6'10"  (2.08m)
Sentence:  4 years
```

```
Variable:  twitch
Name:      Bubba
Height:    6'10"  (2.08m)
Sentence:  4 years
```


Instantiating `String`s with the `new` Keyword

Using the `new` keyword creates two different references to two different objects.

```
String s1 = new String("Test");
String s2 = new String("Test");
```

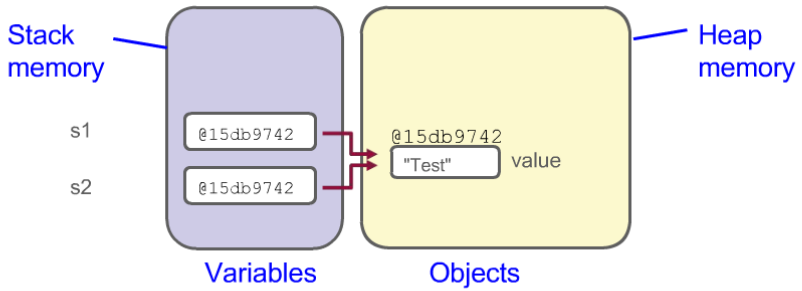


Demo: Curso Java Foundations

Instantiating `Strings` Without the `new` Keyword

- Java automatically recognizes identical `Strings` and saves memory by storing the object only once.
- This creates two different references to one object.

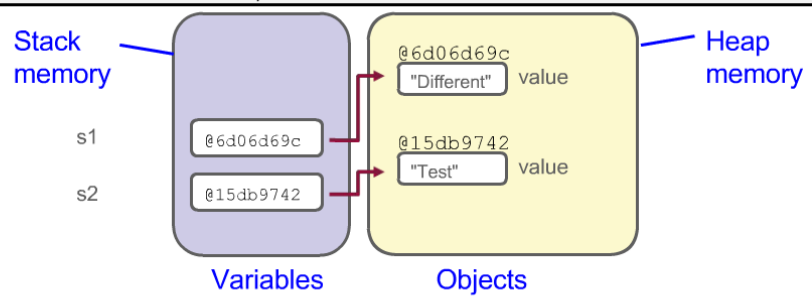
```
String s1 = "Test";
String s2 = "Test";
```



`String` References

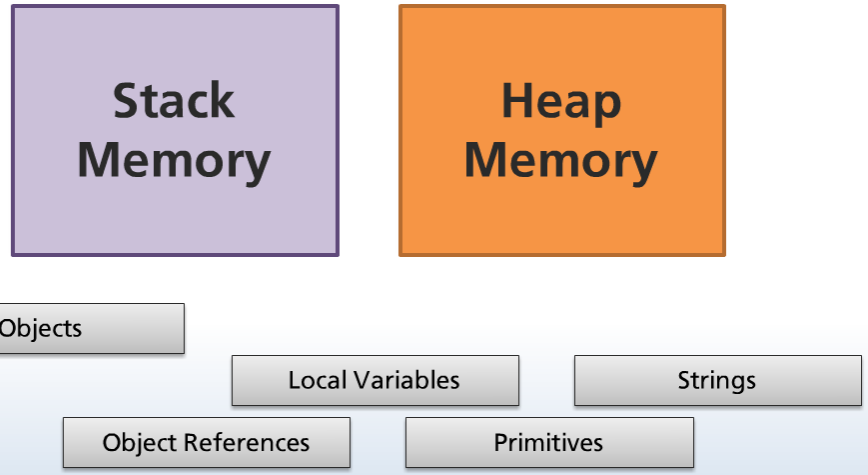
- Altering a `String` using one reference won't affect other references.
- Java allocates new memory for a different `String`.

```
String s1 = "Test";
String s2 = "Test";
s1 = "Different";
```



Demo: Curso Java Foundations

Where would each of the following be stored in memory? Drag each item to the correct box.



What could potentially be the output of the following code?

```
12 Prisoner bubba = new Prisoner();
13 Prisoner twitch = bubba;
14
15 bubba.name = "Bubba";
16 twitch.name = "Twitch";
17
18 System.out.println(bubba.name);
```

- ○ Bubba
- ◉ Twitch
- ○ prisontest.Prisoner@6d06d69c
- ○ name

Demo: Curso Java Foundations

What could you write in line 18 to make **garbage collection** occur?

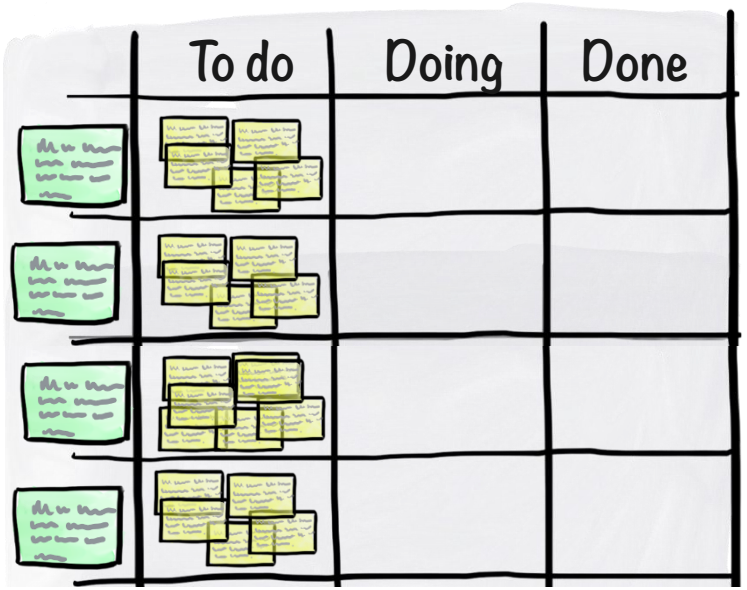
```
12 Prisoner bubba = new Prisoner();
13 Prisoner twitch = new Prisoner();
14
15 bubba.name = "Bubba";
16 twitch.name = "Twitch";
17
18
```

- [x] `bubba = twitch;`
- [x] `bubba = new Prisoner();`
- [ ] `bubba.name = "Twitch";`
- [ ] `bubba == twitch`

According to the following code, how many unique objects are created in the heap memory?

```
String s1 = "Test";
String s2 = "Test";
String s3 = new String("Test");
```

- ○ 0
- ○ 1
- ◉ 2
- ○ 3

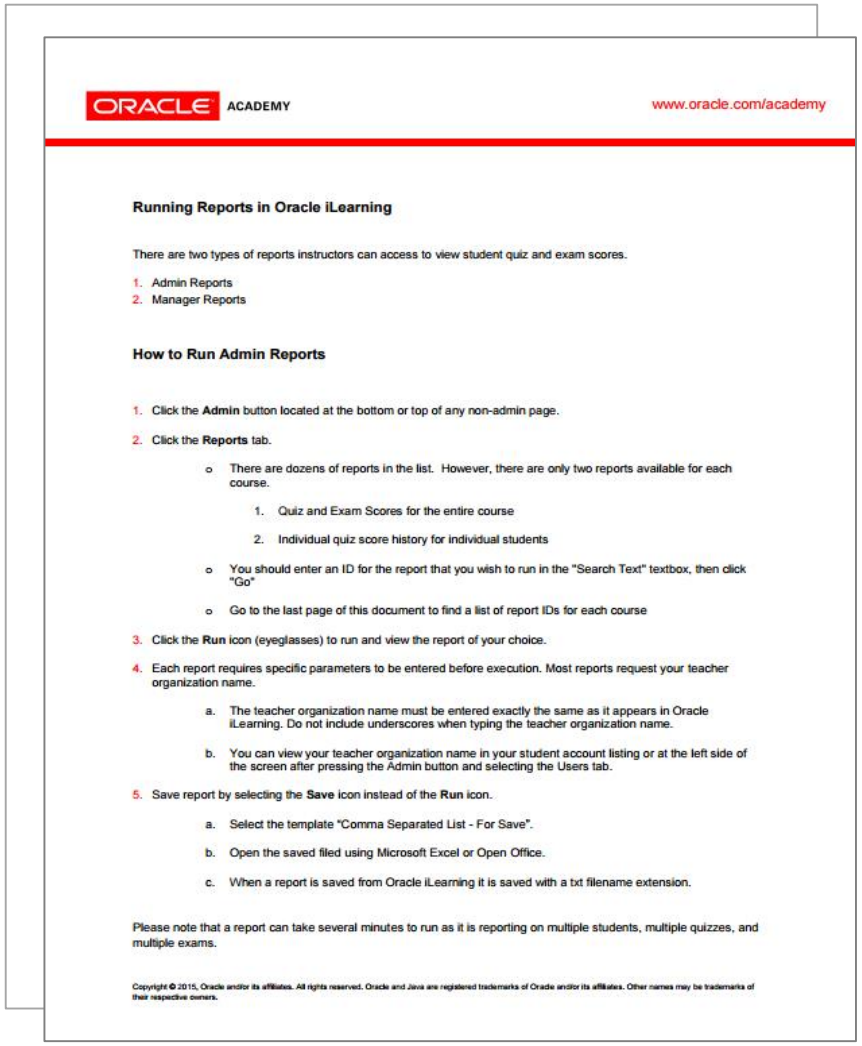


Kanban Board
(Visual Management)

Agenda

1.- Introducción
2.- Plan de formación y certificación en Java
3.- Curso Java Foundations
4.- Estructura del curso Java Foundations
5.- Demo: Curso Java Foundations
6.- Oracle iLearning: Informes y diplomas

Oracle iLearning: Informes y diplomas



ORACLE ACADEMY www.oracle.com/academy

Running Reports in Oracle iLearning

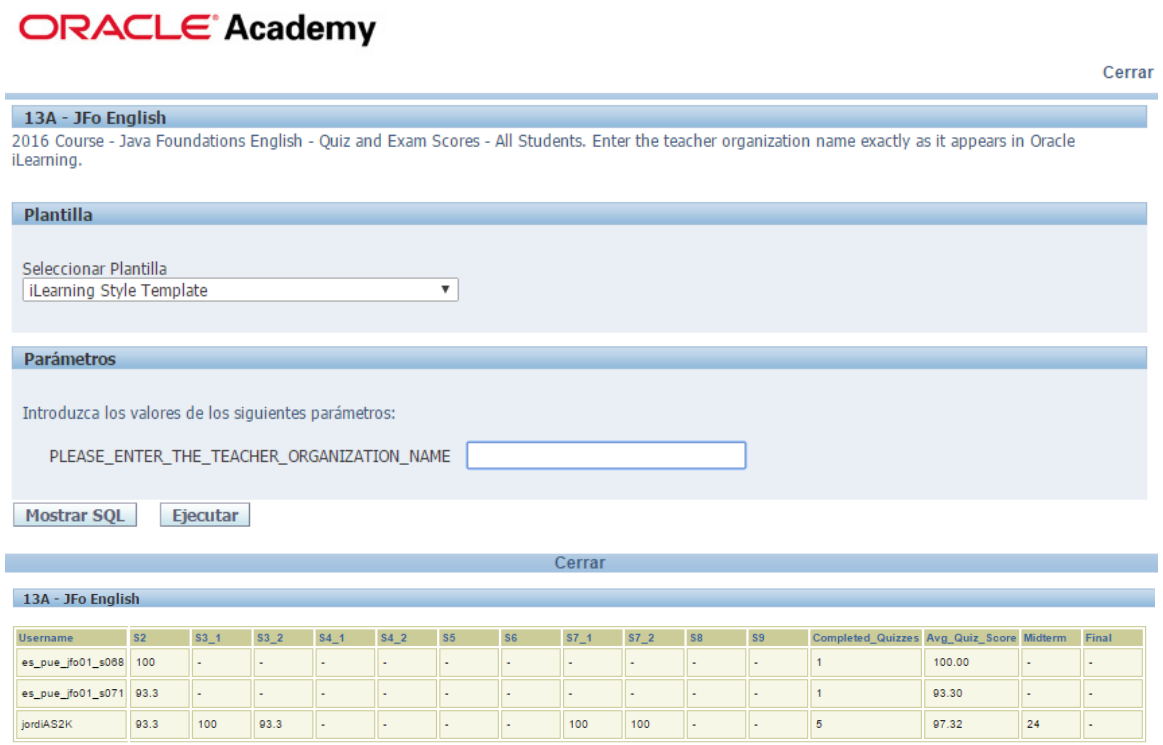
There are two types of reports instructors can access to view student quiz and exam scores.

1. Admin Reports
2. Manager Reports

How to Run Admin Reports

1. Click the **Admin** button located at the bottom or top of any non-admin page.
2. Click the **Reports** tab.
   - There are dozens of reports in the list. However, there are only two reports available for each course.
     1. Quiz and Exam Scores for the entire course
     2. Individual quiz score history for individual students
   - You should enter an ID for the report that you wish to run in the "Search Text" textbox, then click "Go"
   - Go to the last page of this document to find a list of report IDs for each course
3. Click the **Run** icon (eyeglasses) to run and view the report of your choice.
4. Each report requires specific parameters to be entered before execution. Most reports request your teacher organization name.
   a. The teacher organization name must be entered exactly the same as it appears in Oracle iLearning. Do not include underscores when typing the teacher organization name.
   b. You can view your teacher organization name in your student account listing or at the left side of the screen after pressing the Admin button and selecting the Users tab.
5. Save report by selecting the **Save** icon instead of the **Run** icon.
   a. Select the template "Comma Separated List - For Save".
   b. Open the saved filed using Microsoft Excel or Open Office.
   c. When a report is saved from Oracle iLearning it is saved with a txt filename extension.

Please note that a report can take several minutes to run as it is reporting on multiple students, multiple quizzes, and multiple exams.



ORACLE Academy

Cerrar

13A - JFo English

2016 Course - Java Foundations English - Quiz and Exam Scores - All Students. Enter the teacher organization name exactly as it appears in Oracle iLearning.

Plantilla

Seleccionar Plantilla: iLearning Style Template

Parámetros

Introduzca los valores de los siguientes parámetros:

PLEASE_ENTER_THE_TEACHER_ORGANIZATION_NAME

Mostrar SQL | Ejecutar

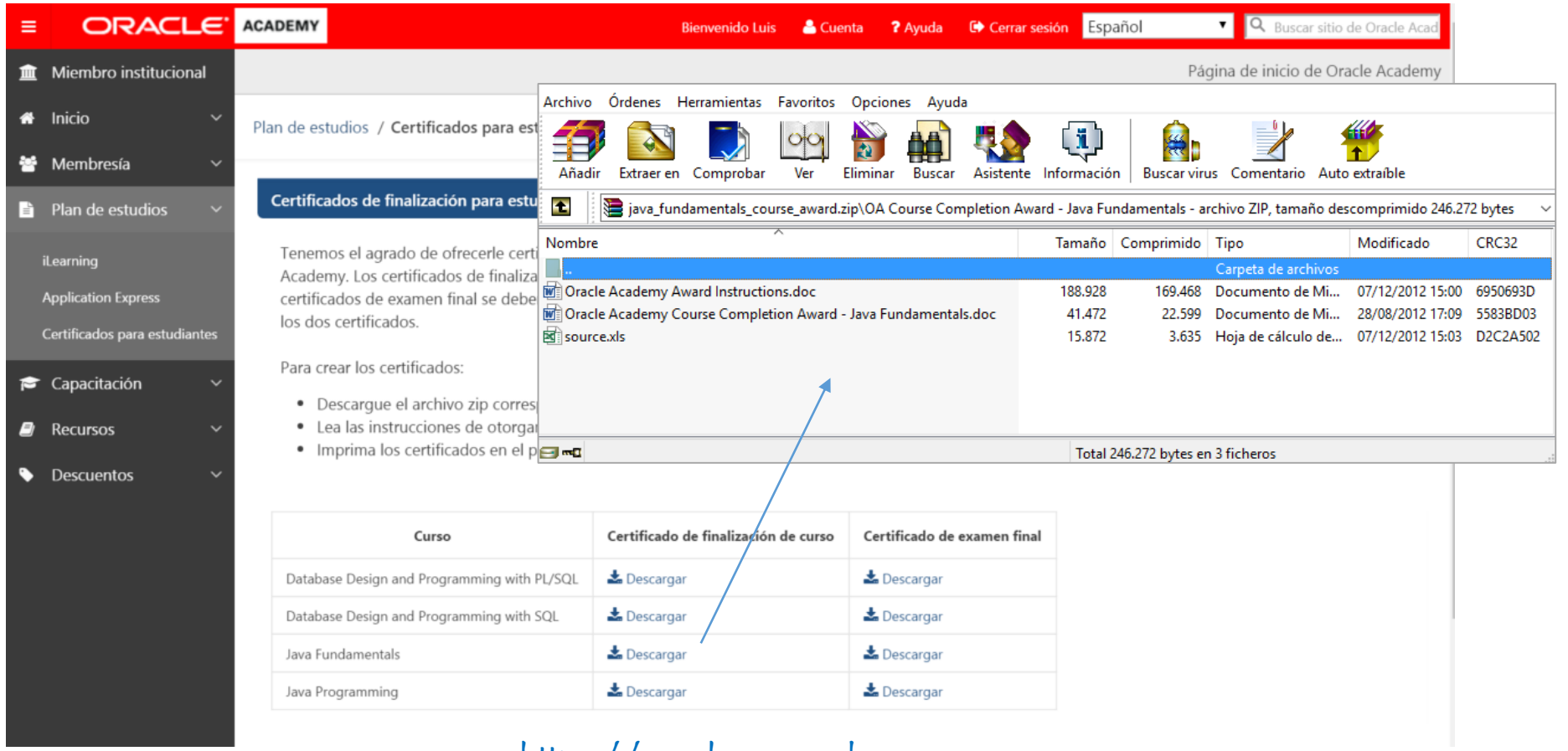
Cerrar

13A - JFo English

Username	S2	S3_1	S3_2	S4_1	S4_2	S5	S6	S7_1	S7_2	S8	S9	Completed_Quizzes	Avg_Quiz_Score	Midterm	Final
es_pue_jfo01_s068	100	-	-	-	-	-	-	-	-	-	-	1	100.00	-	-
es_pue_jfo01_s071	93.3	-	-	-	-	-	-	-	-	-	-	1	93.30	-	-
jordiAS2K	93.3	100	93.3	-	-	-	-	100	100	-	-	5	97.32	24	-

Report Type	Quiz and Exam Scores All Students	Individual Student, Individual Quiz
Report Number	1717763372	1717766196

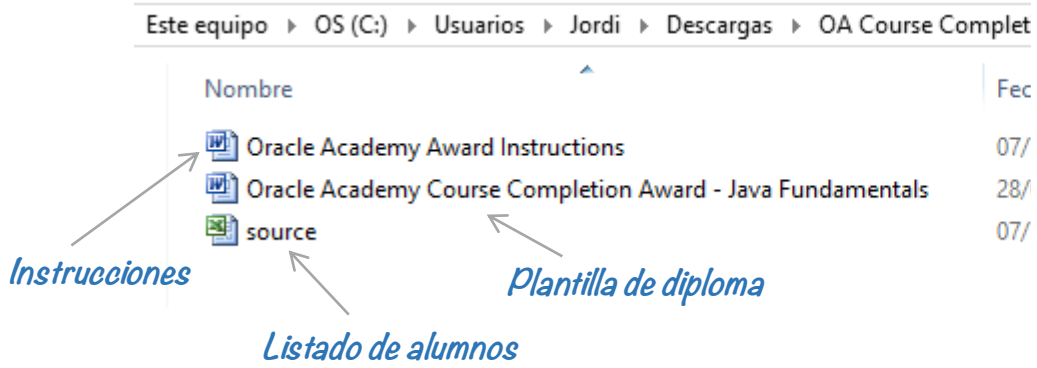
Oracle iLearning: Informes y diplomas



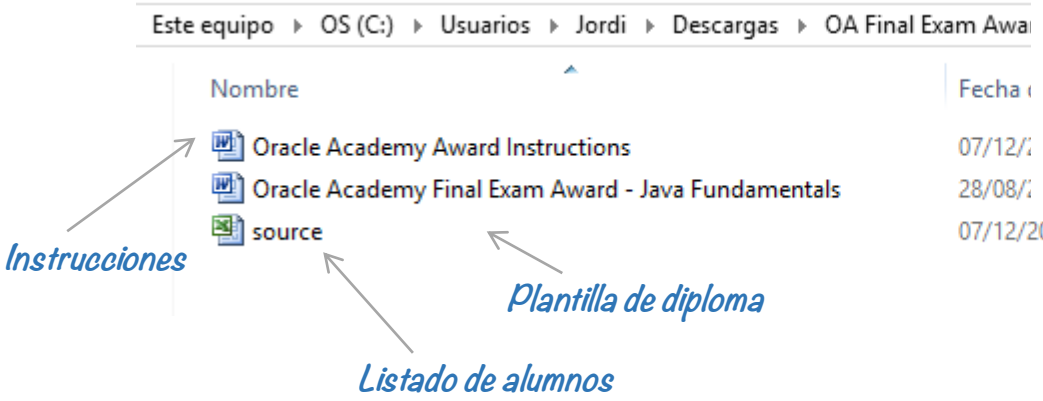
https://academy.oracle.com

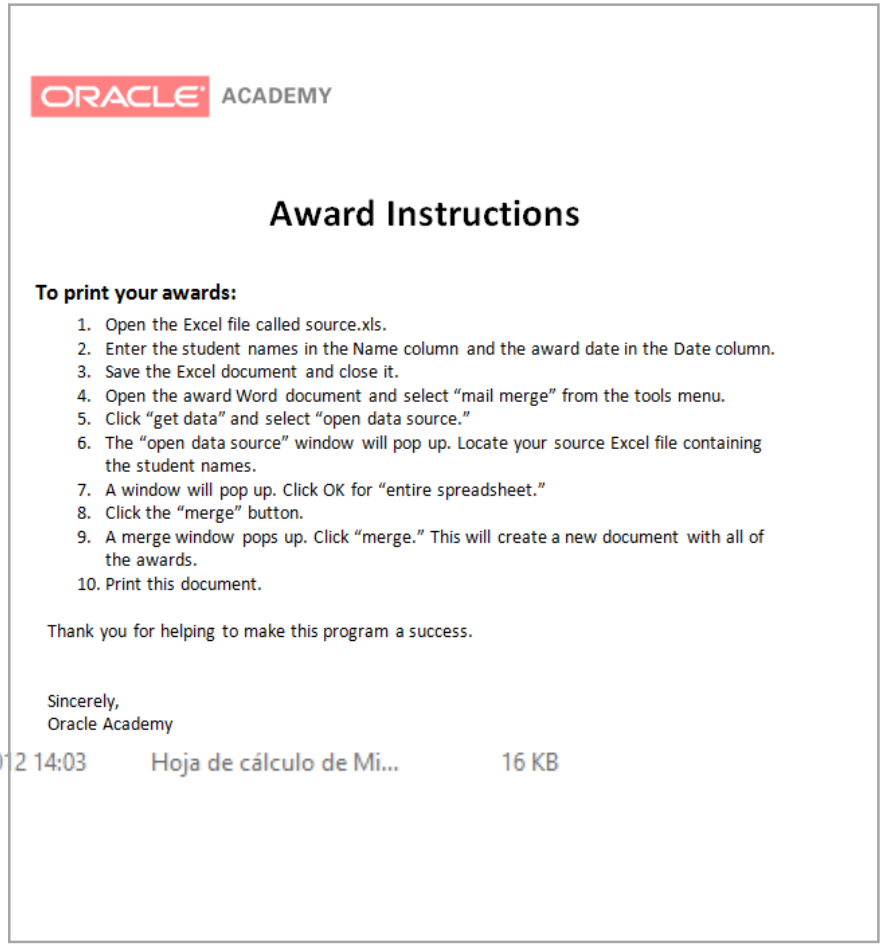
Oracle iLearning: Informes y diplomas

Certificado de finalización del curso



Certificado de superación del examen final





ORACLE ACADEMY

Award Instructions

To print your awards:

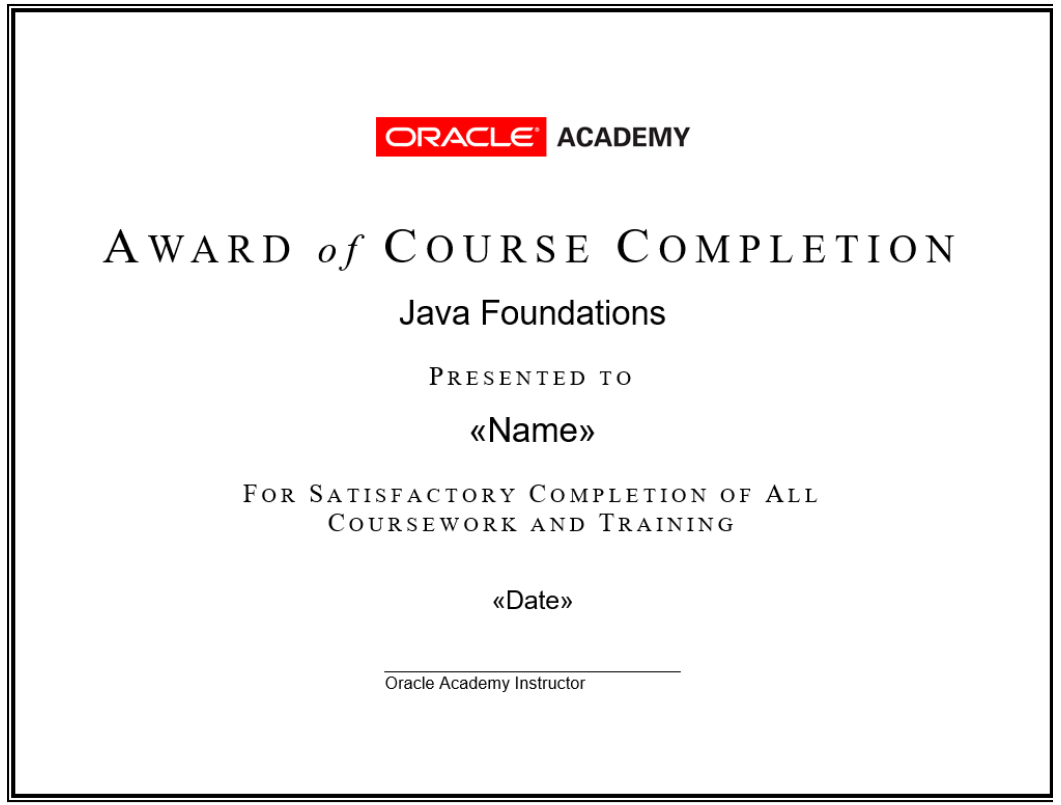
1. Open the Excel file called source.xls.
2. Enter the student names in the Name column and the award date in the Date column.
3. Save the Excel document and close it.
4. Open the award Word document and select "mail merge" from the tools menu.
5. Click "get data" and select "open data source."
6. The "open data source" window will pop up. Locate your source Excel file containing the student names.
7. A window will pop up. Click OK for "entire spreadsheet."
8. Click the "merge" button.
9. A merge window pops up. Click "merge." This will create a new document with all of the awards.
10. Print this document.

Thank you for helping to make this program a success.

Sincerely,
Oracle Academy

Oracle iLearning: Informes y diplomas

Certificado de finalización del curso



ORACLE ACADEMY

AWARD *of* COURSE COMPLETION

Java Foundations

PRESENTED TO

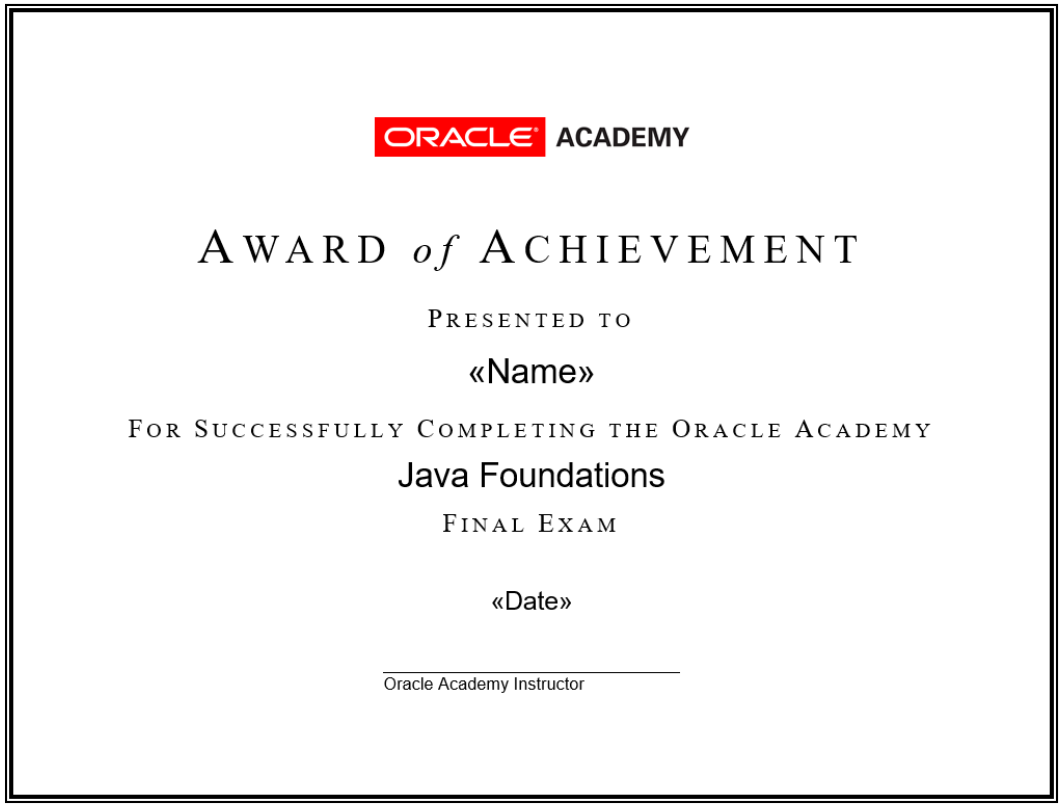
«Name»

FOR SATISFACTORY COMPLETION OF ALL COURSEWORK AND TRAINING

«Date»

Oracle Academy Instructor

Certificado de superación del examen final



ORACLE ACADEMY

AWARD *of* ACHIEVEMENT

PRESENTED TO

«Name»

FOR SUCCESSFULLY COMPLETING THE ORACLE ACADEMY

Java Foundations

FINAL EXAM

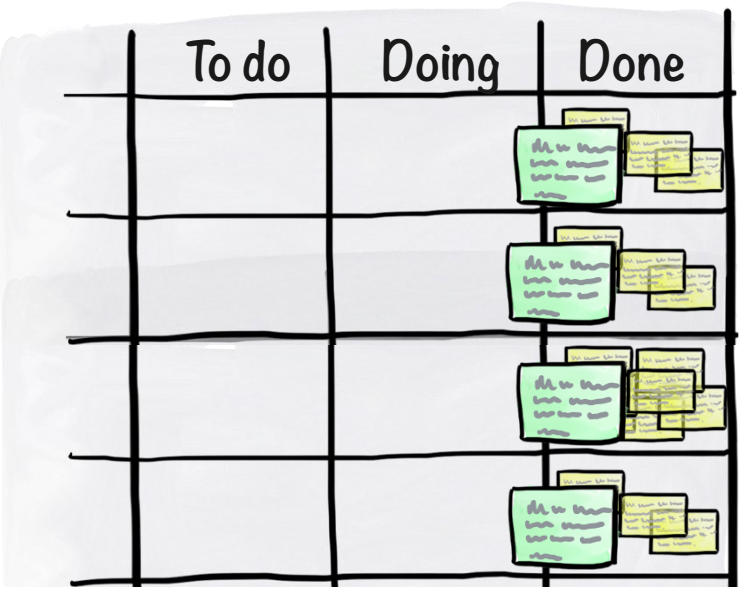
«Date»

Oracle Academy Instructor

¡Muchas gracias!

jordi.arino@pue.es
@jordiAS2K

#beagile



Kanban Board
(Visual Management)

pue

IMPULSANDO EL CONOCIMIENTO
TIC CUALIFICADO

SEDE BARCELONA. Distrito 22@ - Av. Diagonal, 98-100. CP 08019 | T. 93 206 02 49
SEDE MADRID. C/ Arregui y Aruej, 25-27. CP 28007 | T. 91 162 06 69