PLCT S.A de C.V

CURSO JAVA BASICO

Descripción General

Objetivo

Que el estudiante entienda y aplique las funciones, construcciones, librerias y metodologías orientadas a objetos del lenguaje de programación Java

Duración

30 Horas

Requisitos

www.cursoslinux.com.mx
ventas@plct.com.mx
PLCT S.A. de C.V.
Tel.: 55 4522 7839/55 1800 7696/7224447684

CURSO 2023

PLCT S.A de C.V

TEMARIO

1. Breaking the Surface
    1.1. The Way Java Works
    1.2. What you'll do in Java
    1.3. A very brief history of Java
    1.4. Sharpen your pencil
    1.5. Sharpen your pencil answers
    1.6. Code structure in Java
    1.7. Anatomy of a class
    1.8. Writing a class with a main
    1.9. What can you say in the main method?
    1.10. Looping and looping and...
    1.11. there are no Dumb Questions
    1.12. Conditional branching
    1.13. Coding a Serious Business Application
    1.14. Monday morning at Bob's
    1.15. PhraseOMatic
    1.16. Fireside Chats
    1.17. Exercise: Code Magnets
    1.18. Exercise: BE The compiler
    1.19. JavaCross 7.0
    1.20. Mixed Messages
    1.21. Pool Puzzle
    1.22. Exercise Solutins: Code Magnets:
1. A Trip to Objectville
    1.1. Chair Wars
    1.2. Brain Power
    1.3. Making your first object
    1.4. Making and testing Movie objects
    1.5. Quick! Get out of main!
    1.6. Running the Guessing Game
    1.7. Who am I?
1. Know Your Variables
    1.1. Declaring a variable
    1.2. "I'd like a double mocha, no, make it an int."
    1.3. You really don't want to spill that...

www.cursoslinux.com.mx
ventas@plct.com.mx
PLCT S.A. de C.V.
Tel.: 55 4522 7839/55 1800 7696/7224447684

1.4. Back away from that keyword!

1.5. This table reserved.

1.6. Controlling your Dog object

1.7. An object reference is just another variable value.

1.8. There are no Dumb Question

1.9. Java Exposed

1.10. Life on the garbagecollectible heap

1.11. Life and death on the heap

1.12. An array is like a tray of cups

1.13. Arrays are objects too

1.14. Make an array of Dogs

1.15. Control your Dog

1.16. A Dog example

1.17. Exercise: BE the compiler

1.18. Exercise: Code Magnets

1.19. Pool Puzzle

1.20. A Heap

1.21. o' Trouble

1.22. FiveMinute Mystery

1.23. Exercise Solutions: Code Magnets

1.24. Puzzle Solutions

1. How Objects Behave

1.1. Remember: a class describes what an object knows and what an object does

1.2. The size affects the bark

1.3. You can send things to a method

1.4. You can get things back from a method.

1.5. You can send more than one thing to a method

1.6. there are no Dumb Questions

1.7. Reminder: Java cares about type!

1.8. Cool things you can do with parameters and return types

1.9. Encapsulation

1.10. Encapsulating the GoodDog class

1.11. How do objects in an array behave?

1.12. Declaring and initializing instance variables

1.13. The difference between instance and local variables

1.14. there are no Dumb Questions

1.15. Comparing variables (primitives or references)

1.16. Exercise: BE the compiler

1.17. Who am I?

1.18. Mixed Messages

1.19. Pool Puzzle

1.20. Five Minute Mystery

1.21. Puzzle Solutions

1. ExtraStrength Methods

1.1. Let's build a Battleshipstyle game: "Sink a Dot Com"

1.2. First, a highlevel design

1.3. The "Simple Dot Com Game" a gentler introduction

1.4. Developing a Class

1.5. BRAIN POWER

1.6. There are no Dumb Questions

1.7. There are no Dumb Questions

1.8. Exercise: BE the JVM

1.9. Exercise: Code Magnets

1.10. Java Cross

1.11. Exercise Solutions

1. Using the Java Library

1.1. In our last chapter, we left you with the cliffhanger. A bug.

1.2. So what happened?

1.3. How do we fix it?

1.4. Option one is too clunky

1.5. Option two is a little better, but still pretty clunky

1.6. Wake up and smell the library

1.7. Some things you can do with ArrayList

1.8. there are no Dumb Questions

1.9. Java Exposed

1.10. Comparing ArrayList to a regular array

1.11. Comparing ArrayList to a regular array

1.12. Let's fix the DotCom code.

1.13. New and improved DotCom class

1.14. Let's build the REAL game: "Sink a Dot Com"

1.15. What needs to change?

1.16. Who does what in the DotComBust game (and when)

1.17. Prep code for the real DotComBust class

1.18. The final version of the Dotcom class

1.19. Super Powerful Boolean Expressions

1.20. Readybake Code

1.21. Readybake Code

1.22. Using the Library (the Java API)

1.23. You have to know the full name of the class you want to use in your code.

1.24. there are no Dumb Questions

1.25. there are no Dumb Questions

1.26. How to play with the API

1.27. Code Magnets

1.28. JavaCross 7.0

1.29. Exercise Solutions

1.30. JavaCross answers

1. Better Living in Objectville

1.1. Chair Wars Revisited...

1.2. BRAIN POWER

1.3. there are no Dumb Questions

1.4. there are no Dumb Questions

1.5. brain power

1.6. there are no Dumb Questions

1.7. Exercise: Mixed Messages

1.8. Exercise BE the Compiler

1.9. Exercise Solutions: BE the Compiler

1. Serious Polymorphism

1.1. Did we forget about something when we designed this?

1.2. BRAIN POWER

1.3. there are no Dumb Questions

1.4. Pool Puzzle

1.5. Exercise Solutions

1. Life and Death of an Object

1.1. The Stack and the Heap: where things live

1.2. Methods are stacked

1.3. What about local variables that are objects?

1.4. there are no Dumb Questions

1.5. If local variables live on the stack, where do instance variables live?

1.6. The miracle of object creation

1.7. Construct a Duck

1.8. Initializing the state of a new Duck

1.9. there are no Dumb Questions

1.10. Using the constructor to initialize important Duck state

1.11. Make it easy to make a Duck

1.12. Doesn't the compiler always make a no-arg constructor for you? No!

1.13. there are no Dumb Questions

1.14. there are no Dumb Questions

1.15. Wait a minute... we never DID talk about superclasses and inheritance and how that all fits in with

1.16. constructors.

1.17. Making a Hippo means making the Animal and Object parts too...

1.18. How do you invoke a superclass constructor?

1.19. Can the child exist before the parents?

1.20. Superclass constructors with arguments

1.21. Invoking one overloaded constructor from another

1.22. Now we know how an object is born, but how long does an object live?

1.23. What about reference variables?

1. Numbers Matter

1.1. MATH methods: as close as you'll ever get to a global method

1.2. The difference between regular (non-static) and static methods

1.3. What it means to have a class with static methods.

1.4. Static methods can't use nonstatic (instance) variables!

1.5. Static methods can't use nonstatic methods, either!

1.6. Static variable: value is the same for ALL instances of the class

1.7. Initializing a static variable

1.8. static final variables are constants

1.9. final isn't just for static variables...

1.10. there are no Dumb Questions

1.11. Math methods

1.12. Wrapping a primitive

1.13. Before Java 5.0, YOU had to do the work...

1.14. Autoboxing: blurring the line bet ween primitive and object

1.15. Autoboxing works almost everywhere

1.16. But wait! There's more! Wrappers have static utility methods too!

1.17. And now in reverse... turning a primitive number into a String

1.18. Number formatting

1.19. Formatting deconstructed...

1.20. The percent (%) says, "insert argument here" (and format it using these instructions)

1.21. The format String uses its own little language syntax

1.22. The format specifier

1.23. The only required specifier is for TYPE

1.24. What happens if I have more than one argument?

1.25. So much for numbers, what about dates?

1.26. Working with Dates

1.27. Moving backward and for ward in time

1.28. Getting an object that extends Calendar

1.29. Working with Calendar objects

1.30. Highlights of the Calendar API

1.31. Even more Statics!... static imports

1.32. Lunar Code Magnets