

## CURSO JAVA AVANZADO

### OBJETIVO

Que el estudiante entienda y aplique las funciones, construcciones, librerías para el desarrollo y distribución de aplicaciones distribuidas y con interfaz gráfica.

### TEMARIO

- **Risky Behavior**
  - Let's make a Music Machine
  - We'll start with the basics
  - First we need a Sequencer
  - Something's wrong!
  - What happens when a method you want to call (probably in a class you didn't write) is risky?
  - Methods in Java use exceptions to tell the calling code, "Some thing Bad Happened. I failed."
  - The compiler needs to know that YOU know you're calling a risky method.
  - An exception is an object... of type Exception.
  - If it's your code that catches the exception, then whose code throws it?
  - there are no Dumb Questions
  - Flow control in try/catch blocks
  - Finally: for the things you want to do no matter what.
  - Sharpen your pencil:Flow Control
- Did we mention that a method can throw more than one exception?
- Exceptions are polymorphic
- Multiple catch blocks must be ordered from smallest to biggest
- You can't put bigger baskets above smaller baskets.
- Sharpen your pencil
- When you don't want to handle an exception...
- Ducking (by declaring) only delays the inevitable
- Getting back to our music code...
- Code Kitchen
- Making actual sound
- Your very first sound player app
- Making a MidiEvent (song data)
- MIDI message: the heart of a MidiEvent
- Anatomy of a message
- Version 2: Using command-line args to experiment with sounds
- Exercise: True or False
- Exercise: Code Magnets
- JavaCross 7.0



- Exercise Solutions: True or False
- Code Magnets
- JavaCross Answers
- **A Very Graphic**
  - It all starts with a window
- Your first GUI: a button on a frame Ver en [HTML](#) o en [PDF](#) Calendario: Ver en [PDF](#)  
 Requisitos: Conocimientos básicos de Linux o conocimientos equivalentes  
 Fecha de inicio: No hay fecha definida  
 Fecha de fin: No hay fecha definida  
 Duración: 30 horas.  
 Horario: 09:00 a.m. a 3:00 p.m.  
 Material incluido: Manual Oficial y CD's de la distrie
  - Getting a user event
  - Getting a button's ActionEvent
  - There are no Dumb Questions
  - Make your own drawing widget
  - Because life's too short to paint the circle a solid color when there's a gradient blend waiting for you
  - BULLET POINTS
  - We can get an event.
  - Let's try it with TWO buttons
  - Java Exposed: This weeks interview: Instance of an Inner Class
  - There are no Dumb Questions
  - Code Kitchen
  - Exercise: Who am I?
  - Exercise: BE the compiler
  - Pool Puzzle
  - Exercise Solutions: Who am I?
  - Pool Puzzle
- **Work on Your Swing**
  - Swing components
  - Layout Managers
  - How does the layout manager decide?
  - The Big Three layout managers: border, flow, and box.
- there are no Dumb Questions
- Playing with Swing components
- there are no Dumb Questions
- Code Kitchen
- Making the BeatBox
- Exercise: Which code goes with which layout?
- Code Fragments
- **Saving Objects**
  - Capture the Beat
  - Saving State
  - Writing a serialized object to a file
  - Data moves in streams from one place to another
  - What really happens to an object when it's serialized?
  - But what exactly IS an object's state? What needs to be saved?
  - If you want your class to be serializable, implement Serializable
  - There are no Dumb Questions
  - Deserialization: restoring an object
  - What happens during deserialization?
  - There are no Dumb Questions
  - Saving and restoring the game characters
  - The GameCharacter class
  - Object Serialization
  - Writing a String to a Text File
  - Text File Example: e-Flashcards
  - Quiz Card Builder (code outline)
  - The java.io.File class
  - The beauty of buffers
  - Reading from a Text File
  - Quiz Card Player (code outline)
  - Parsing with String split()
  - There are no Dumb Questions
  - Version ID: A Big Serialization Gotcha
  - Using the serialVersionUID
  - Code Kitchen



- Saving a BeatBox pattern
- Restoring a BeatBox pattern
- Sharpen your pencil: Can they be saved?
- Exercise: True or False
- Code Magnets
- Exercise Solutions
- **Make a Connection**
  - Real-time Beat Box Chat
  - Connecting, Sending, and Receiving
  - Make a network Socket connection
  - A TCP port is just a number: A 16-bit number that identifies a specific program on the server.
  - To read data from a Socket, use a BufferedReader
  - To write data to a Socket, use a PrintWriter
  - DailyAdviceClient code
  - Writing a simple server
  - DailyAdviceServer code
  - Writing a Chat Client
  - Java has multiple threads but only one Thread class
  - What does it mean to have more than one call stack?
  - Every Thread needs a job to do: A method to put on the new thread stack.
  - To make a job for your thread, implement the Runnable interface
  - The Thread Scheduler
  - there are no Dumb Questions
  - Putting a thread to sleep
  - Using sleep to make our program more predictable.
  - Making and starting two threads
  - What will happen?
  - Um, yes. There IS a dark side. Threads can lead to concurrency
- 'issues'.
  - The Ryan and Monica problem, in code
  - The Ryan and Monica example
  - We need the makeWithdrawal ( ) method to run as one atomic thing.
  - Using an object's lock
  - The dreaded "Lost Update" problem
  - Let's run this code...
  - Make the increment() method atomic. Synchronize it!
  - there are no Dumb Questions
  - The deadly side of synchronization
  - New and improved SimpleChatClient
  - Ready-bake Code: The really really simple Chat Server
  - there are no Dumb Questions
  - Code Kitchen
  - Exercise: Code Magnets
  - Exercise Solutions
  - Minute Mystery
- **Data structures**
  - Tracking song popularity on your jukebox
  - Here's what you have so far, without the sort:
  - But the ArrayList class does NOT have a sort() method!
  - ArrayList is not the only collection
  - You could use a TreeSet... Or you could use the Collections.sort() method
  - Adding Collections.sort() to the Jukebox code
  - But now you need Song objects, not just simple Strings.
  - Changing the Jukebox code to use Songs instead of Strings
  - It won't compile!
  - The sort() method declaration



- Generics means more type-safety
- Learning generics
- Using generic CLASSES
- Using type parameters with ArrayList
- Using generic METHODS
- Here's where it gets weird...
- Revisiting the sort() method
- In generics, "extends" means "extends or implements"
- Finally we know what's wrong...
- The new, improved, comparable Song class
- We can sort the list, but...
- Using a custom comparator
- Updating the Jukebox to use a Comparator
- Sharpen your pencil: Reverse Engineer
- Sharpen your pencil: Fill-in-the-blanks
- Uh-oh. The sorting all works, but now we have duplicates...
- We need a Set instead of a List
- The Collection API (part of it)
- Using a HashSet instead of ArrayList
- What makes two objects equal?
- How a HashSet checks for duplicates: hashCode() and equals()
- The Song class with overridden hashCode() and equals()
- there are no Dumb Questions
- And if we want the set to stay sorted, we've got TreeSet
- What you MUST know about TreeSet...
- TreeSet elements MUST be comparable
- We've seen Lists and Sets, now we'll use a Map

- Finally, back to generics
- Using polymorphic arguments and generics
- But will it work with ArrayList<Dog> ?
- What could happen if it were allowed...
- Wildcards to the rescue
- Alternate syntax for doing the same thing
- there are no Dumb Questions
- Exercise: BE the compiler, advanced
- Solution to the "Reverse Engineer" sharpen exercise
- Exercise Solution
- BE the compiler solution
- **Release Your Code**
  - Deploying your application
  - Imagine this scenario...
  - Separate source code and class files
  - Put your Java in a JAR
  - Running (executing) the JAR
  - Put your classes in packages!
  - Preventing package name conflicts
  - Compiling and running with packages
  - The -d flag is even cooler than we said
  - Making an executable JAR with packages
  - So where did the manifest file go?
  - Java Web Start
  - The .jnlp file
  - Steps for making and deploying a Java Web Start app
  - What's First?
  - True or False
  - Summary-Cross 7.0
- **Distributed Computing**



- Method calls are always between two objects on the same heap.
- What if you want to invoke a method on an object running on another machine?
- Object A, running on Little, wants to call a method on Object B, running on Big
- But you can't do that
- The role of the 'helpers'
- Java RMI gives you the client and service helper objects!
- How does the client get the stub object?
- How does the client get the stub class?
- Be sure each machine has the class files it needs.
- Sharpen your pencil: What's First?
- Yeah, but who really uses RMI?

- What about Servlets?
- A very simple Servlet
- HTML page with a link to this servlet
- There are no Dumb Questions
- Just for fun, let's make the Phrase-O-Matic work as a servlet
- Phrase-O-Matic code, servlet-friendly
- Enterprise JavaBeans: RMI on steroids
- For our final trick... a little Jini
- Adaptive discovery in action
- Self-healing network in action
- Final Project: the Universal Service browser
- How it works:
- The classes and interfaces:
- Sharpen your pencil
- Congratulations!

