# JAVA FOUNDATIONS 1Z0-811

ORACLE ACADEMY

2 DE SEPTIEMBRE DE 2025

https://academy.oracle.com/

HTTPS://GITHUB.COM/ISC-UPA/2025-3-TIID3C-POO

# Contenido

→

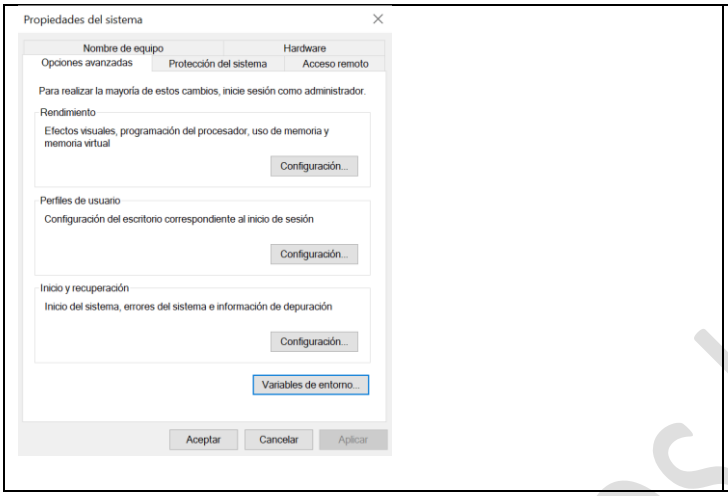
# 1. Introduction

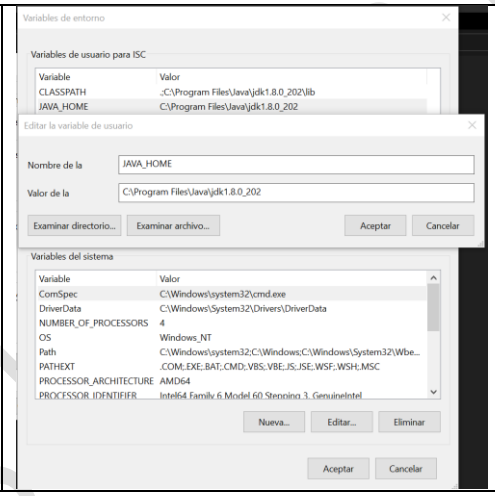
## 1.1. Technological Requirements:

Java JDK https://www.oracle.com/java/technologies/downloads/ jdk-8u202-windows-x64.exe
VS Code https://code.visualstudio.com/Download VSCodeSetup-x64-1.103.2.exe
Extensions: **Extension Pack for Java**

Integrated Development Environment (IDE)
Eclipse IDE: https://www.eclipse.org/downloads/packages/
NetBeans IDE https://netbeans.apache.org/download/index.html

Variables de entorno




Panel de control -> Sistema -> Configuracion avanzada del sistema
Opciones avanzadas -> Variables de entorno -> Variables de Usuario

| | |
|---|---|
| JAVA_HOME<br>C:\Program Files\Java\jdk1.8.0_202 | PATH<br>%JAVA_HOME%\BIN |
| CLASSPATH<br>**.;** %JAVA_HOME%\LIB | Probar Instalación desde CMD<br>C:\>java -version (correr)<br>C:\>javac -version (compilar) |

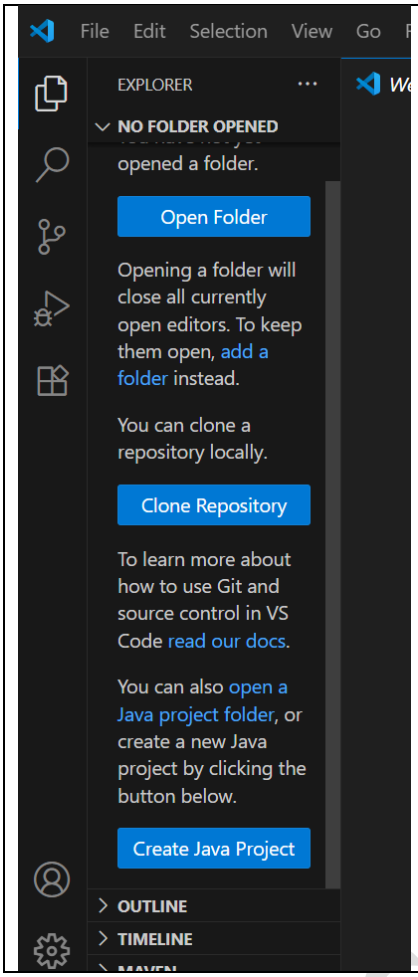
```
C:\dev>java -version
java version "1.8.0_202"

C:\dev>javac -version
javac 1.8.0_202

C:\dev\poo>javac Hola.java

C:\dev\poo>java Hola
Hello World!
```

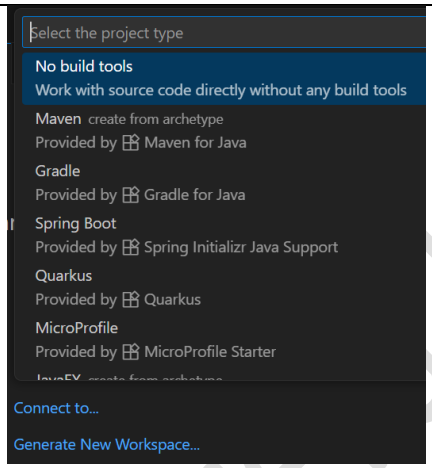
```
public class Hola {

    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

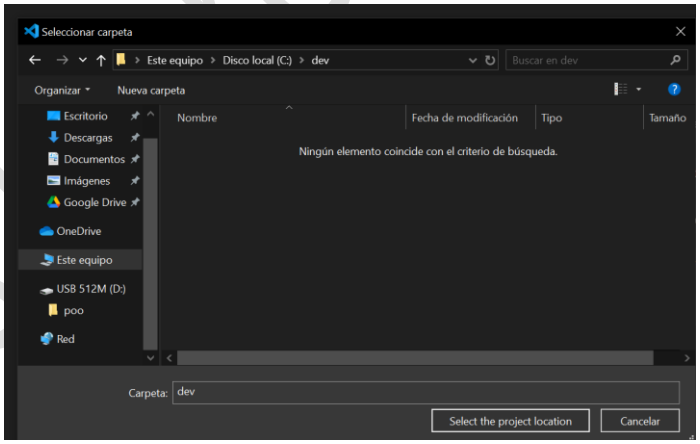
→

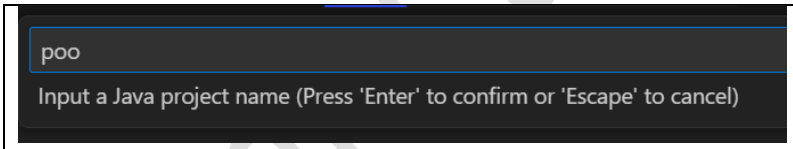
Crear un Directorio de trabajo
C:\> mkdir dev
C:\dev>

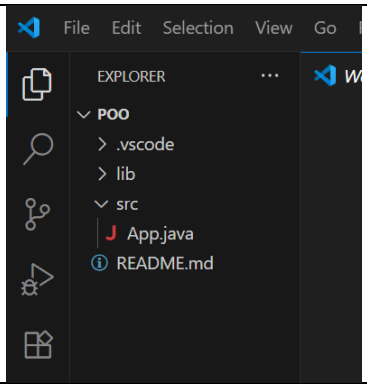
## 1.2. Create Java Project:



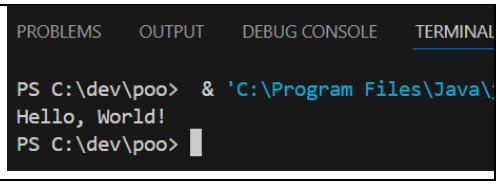







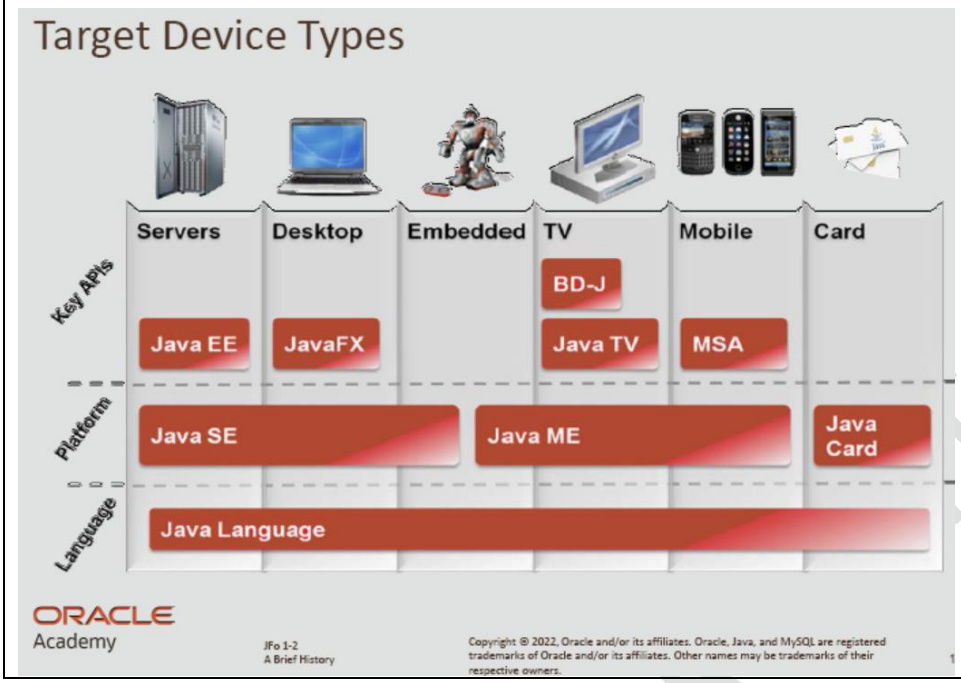

→

## 1.3. Setting Up Java

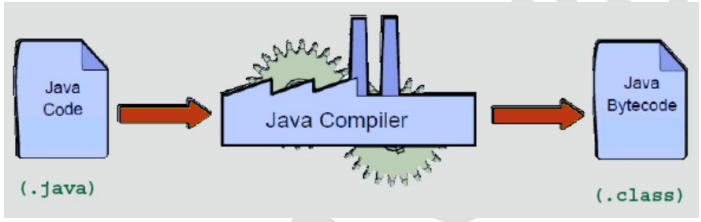
James Gosling is considered the “Father of Java”. Duke, the Java Mascot.

Oracle acquired Sun Microsystems in 2010, and released JDK 7 in 2011, and JDK 8 in 2014.

Jakarta EE Is used to create large enterprise, server-side, and client-side distributed applications

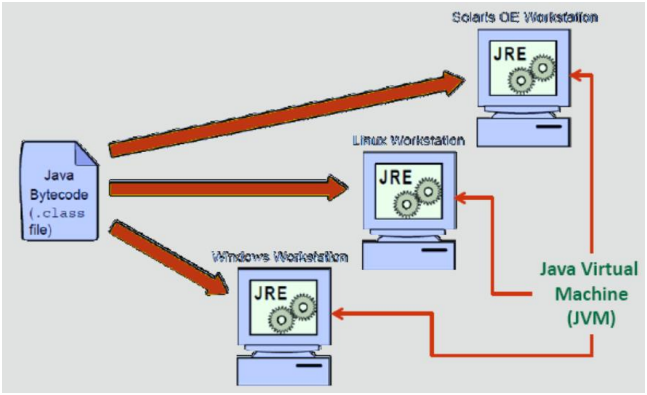



Java is Platform-Independent

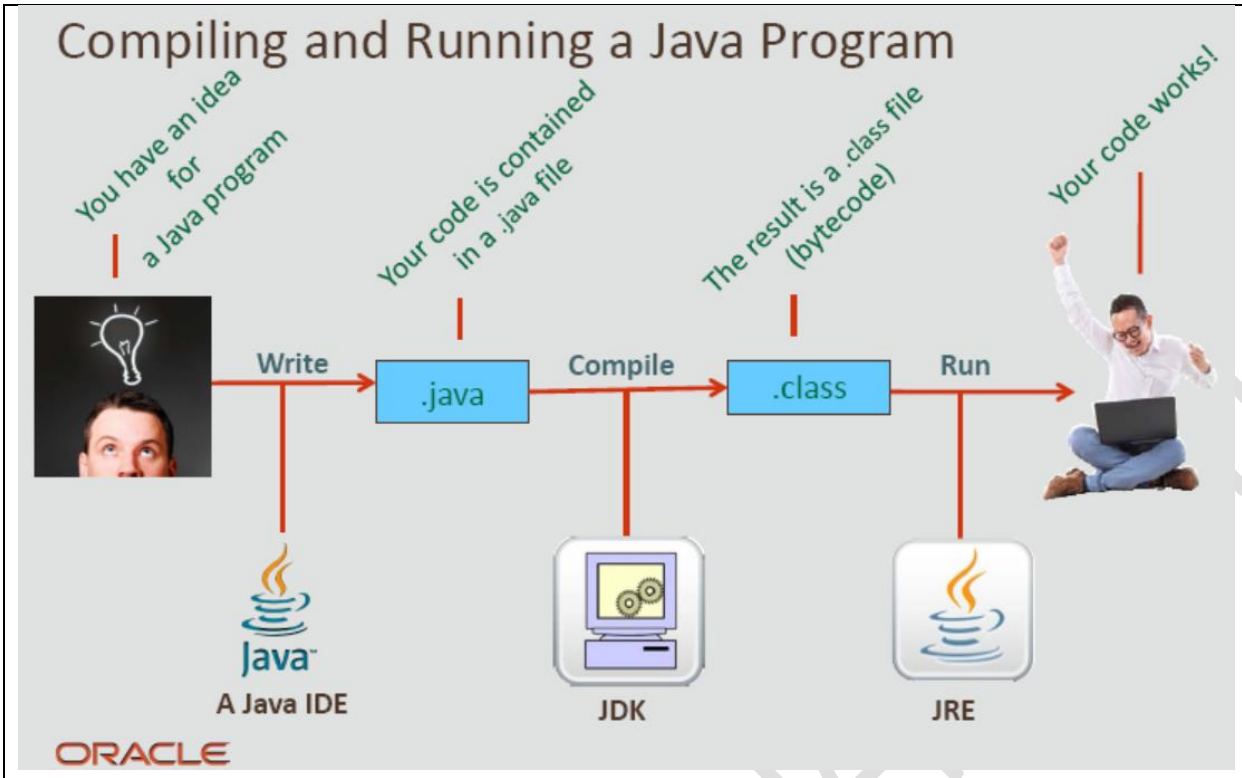


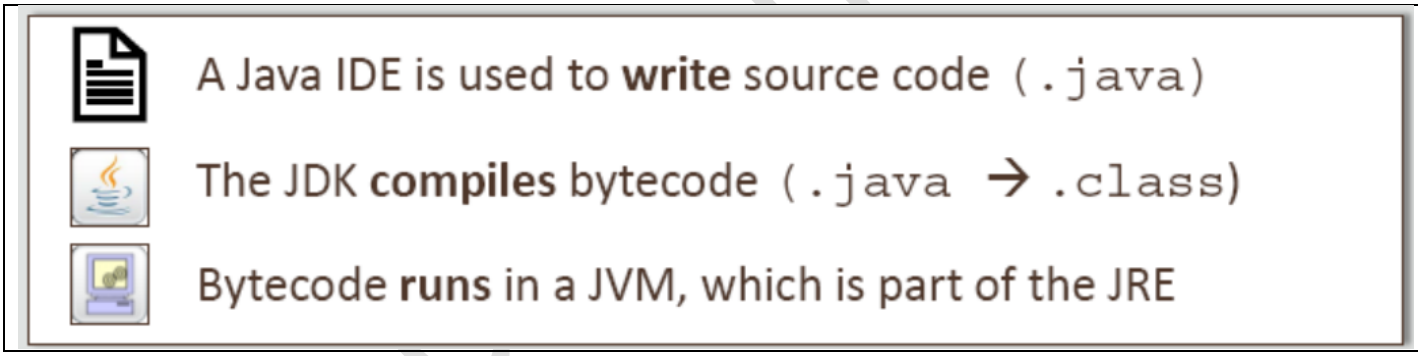

- 

Java Programs Run in a JVM




| Java Runtime Environment (JRE) | Java Development Kit (JDK) |
|---|---|
| Includes:<br>• The Java Virtual Machine (JVM)<br>• Java class libraries<br>Purpose:<br>• Read bytecode (.class)<br>• Run the same bytecode anywhere with a JVM | Includes:<br>• JRE Java Compiler<br>• Additional tools<br>Purpose:<br>Compile bytecode (.java □.class) |


Compiling and Running a Java Program
You have an idea for a Java program
Your code is contained in a .java file
The result is a .class file (bytecode)
Your code works!
Write
.java
Compile
.class
Run
Java
A Java IDE
JDK
JRE
ORACLE

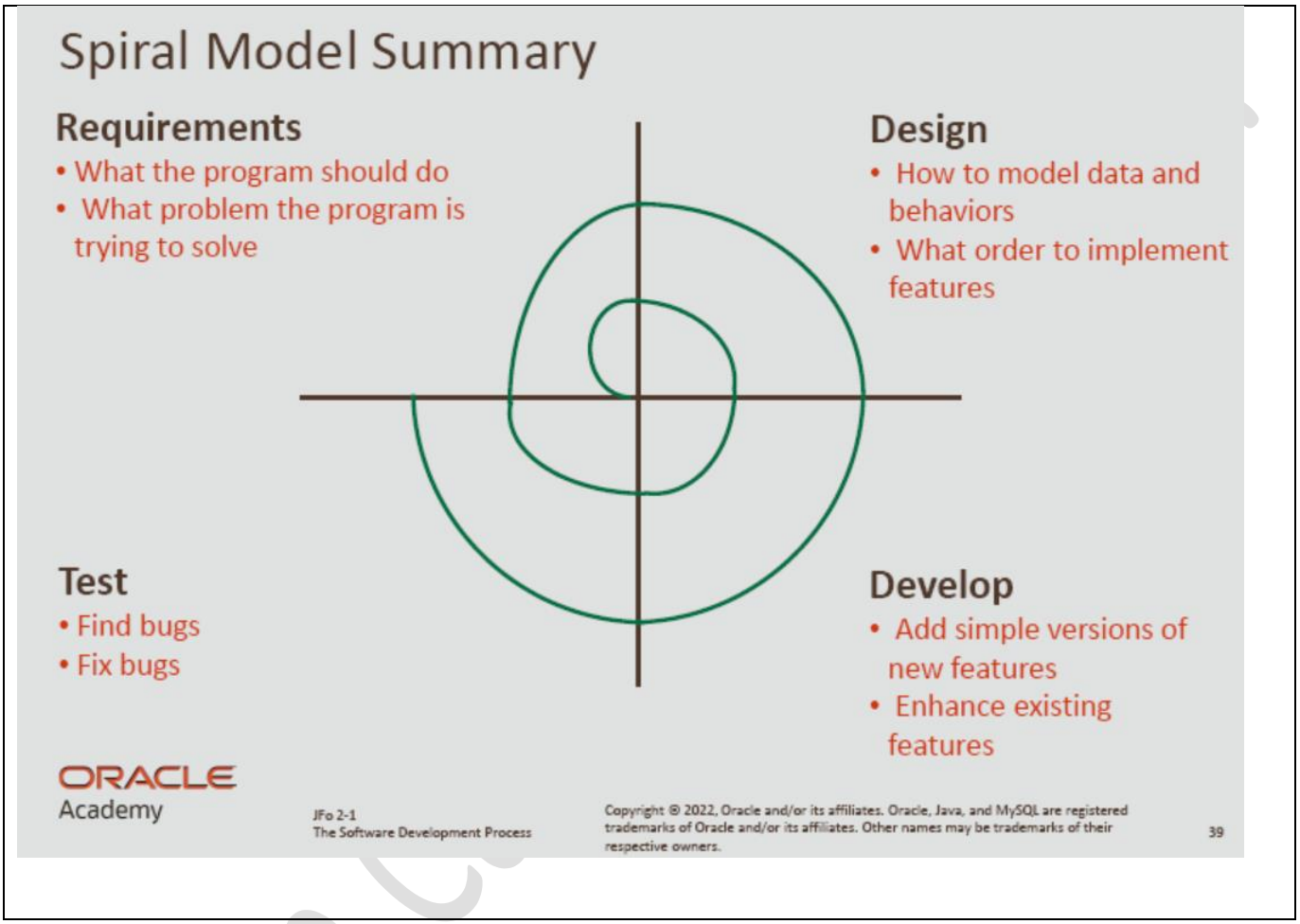

A Java IDE is used to write source code (.java)
The JDK compiles bytecode (.java → .class)
Bytecode runs in a JVM, which is part of the JRE

→

# 2. Java Basics

## 2.1. The Software Development Process

Spiral Model of Development




https://objectstorage.uk-london-1.oraclecloud.com/n/lrvrlgaqj8dd/b/Games/o/JavaPuzzleBall/index.html

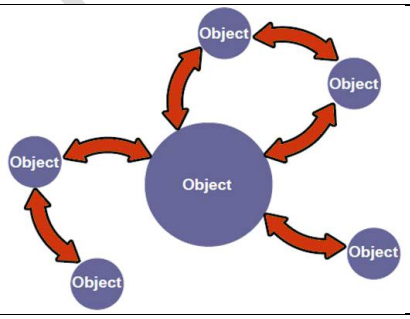
→

## 2.2. What is my Program Doing?

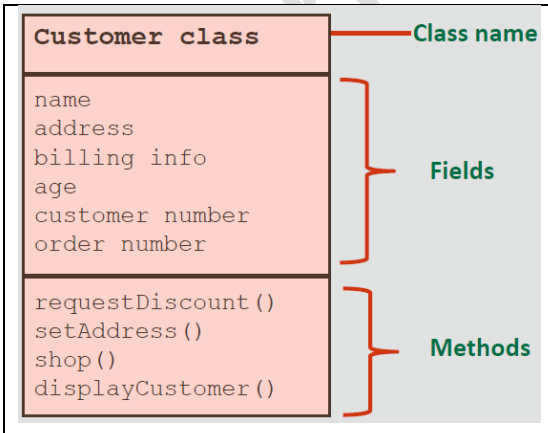
| | |
|---|---|
| Code within curly braces is called a block of code<br>Indentation before a line of code (4 spaces)<br>Whitespace<br>End statements with semicolons (;)<br><br>// Single-line comments<br><br>Multi-line comments<br>/* Bievenidos<br>a poo<br>*/ | public static void NombreMetodo() { . . }<br>NombreMetodo(); // llamar al método<br><br>Debug<br>To set a breakpoint<br>Press Step Over |

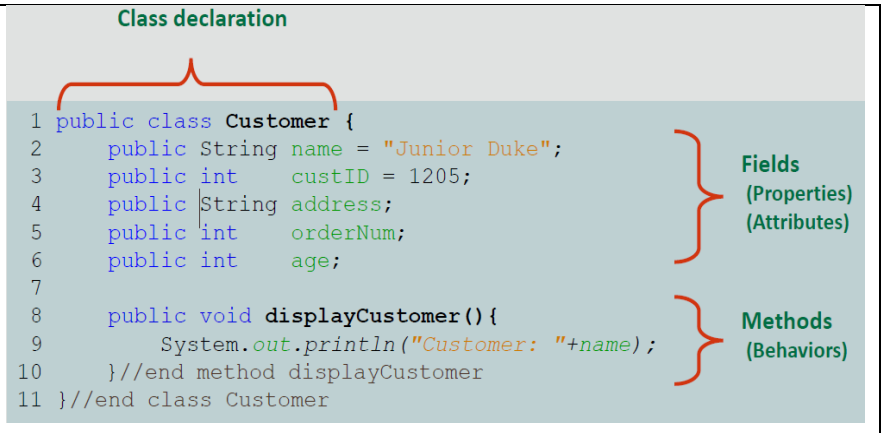
## 2.3. Introduction to Object-Oriented Programming Concepts

| | |
|---|---|
| Procedural languages …<br>• Read one line at a time<br>• The C language is procedural | Object-oriented languages…<br>• Read one line at a time<br>• Model objects through code<br>• Emphasize object interaction<br>• Allow interaction without a prescribed order<br>• Java and C++ are object-oriented languages |

| | |
|---|---|
| Object-Oriented Programming<br>• Interaction of objects<br>• No prescribed sequence |  |

Modeling Properties and Behaviors




**Quiz: JFo - Section 2 Questions 15**

→

# 3. Java Data Types

## 3.1. What is a Variable?

String x ="Sam";
System.out.println("My name is " + x);  Variables03.java (There are 6 mistakes)

| Type | Keyword | Example Values |
|---|---|---|
| Boolean | boolean | true, false |
| Integer | int | 1, -10, 20000, 123_456_789 |
| Double | double | 1.0, -10.0005, 3.141 |
| String | String | "Alex", "I ate too much dinner." |

**Variable Naming Conventions**

- Begin each variable with a lowercase letter
- Subsequent words should be capitalized: myVariable
- Choose names that are mnemonic and that indicate the intent of the variable to the casual observer
- Remember that …
- Names are case-sensitive
- Names can't include white space

Int studentAge = 20;
String myCatchPhrase = "Enjoy Alex Appreciation Day!";

## 3.2. Numeric Data

Integral Primitive Types

| Type | Length | Number of Possible Values | Minimum Value | Maximum Value |
|---|---|---|---|---|
| **Byte** | 8 bits | $2^8$, or… 256 | $-2^7$, or… −128 | $2^7-1$, or… 127 |
| **short** | 16 bits | $2^{16}$, or… 65,535 | $-2^{15}$, or… −32,768 | $2^{15}-1$, or… 32,767 |
| **int** | 32 bits | $2^{32}$,or… 4,294,967,296 | $-2^{31}$, or… −2,147,483,648 | $2^{31}-1$, or… 2,147,483,647 |
| **long** | 64 bits | $2^{64}$, or… 18,446,744,073,709,551,616 | $-2^{63}$, or… −9,223,372,036,854,775,808L | $2^{63}-1$, or… 9,223,372,036,854,775,807L |

+=  -=  *=  /=  %=  ++  --  Pre/Post  a+=b  a = a + ( b )

// pre y post incremento y decremento

```
int players = 0;
System.out.println("players online: " + players++);
System.out.println("The value of players is " + players);
System.out.println("The value of players is now " + ++players);
System.out.println("The value of players is " + players);
```

Floating Point Primitive Types

| Type | Float Length | When will I use this? |
|---|---|---|
| `float` | 32 bits | Never |
| `double` | 64 bits | Often |

double x = 9/2;  double x = 9/2.0;

**final** double PI = 3.141592;

Final variable naming conventions:

- Capitalize every letter
- Separate words with an underscore
  MINIMUM_AGE

## Rules of Precedence

- Operators within a pair of parentheses
- Increment and decrement operators (++or --)
- Multiplication and division operators, evaluated from left to right
- Addition and subtraction operators, evaluated from left to right
- If operators of the same precedence appear successively, the operators are evaluated from left to right

```
int x = (((25 - 5) * 4) / (2 - 10)) + 4;
int y = 25 - 5 * 4 / 2 - 10 + 4;
```

→

## 3.3. Textual Data

Use the char data type
Use Strings
Concatenate Strings
Understand escape sequences
Understand print statements better

| | |
|---|---|
| Char is used for a single character (16 bits)<br>char shirtSize= 'M'; | A String can handle multiple characters<br>String greeting = "Hello World!"; // Asignación Hard-coding |

### Primitives

| Type | Length | Data |
|---|---|---|
| boolean | 1 bit | true / false |
| byte | 8 bits | Integers |
| short | 16 bits | Integers |
| int | 32 bits | Integers |
| long | 64 bits | Integers |
| float | 32 bits | Floating point numbers |
| double | 64 bits | Floating point numbers |
| char | 16 bits | Single characters |

**Where are Strings?**

String is capitalized

- Strings are an object, not a primitive
- Object types are capitalized by convention

Combining multiple Strings is called concatenation
String totalPrice = "Total: $" +3 +2 +1;
String totalPrice = 3 +2 + 1 + "Total: $";
String totalPrice = "Total: $" +(3 +2 +1);

### Escape Sequence

| Escape Sequence | Description |
|---|---|
| \t | Insert a new tab |
| \b | Insert a backspace |
| \n | Insert a new line |
| \r | Insert a carriage return |
| \f | Insert a formfeed |
| \' | Insert a single quote character |
| \" | Insert a double quote character |
| \\ | Insert a backslash character |

System.out.print**ln**("The cat said \"Meow!\" to me.");
println() vs. print()

System.out.println("1\t2\t3\t\"Hola\" mundo");
1 2 3 "Hola" mundo

System.out.println("Hola\nAdios");
Hola
Adios

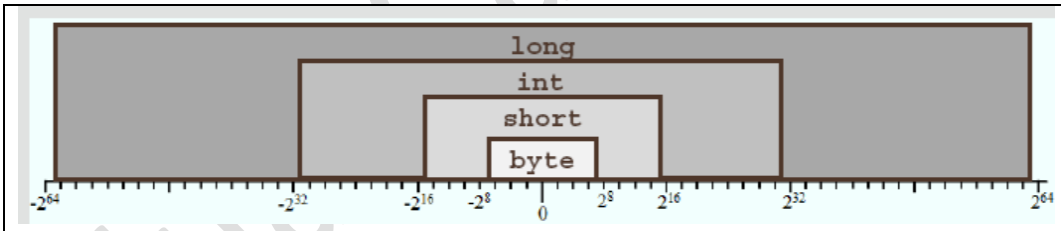
```
System.out.println("This is the first line."
                + "This is NOT the second line.");
```

sout tab "Metodo abreviado"

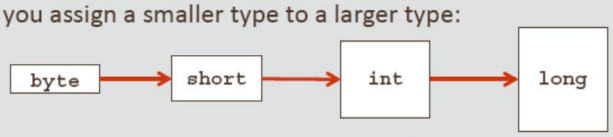
## 3.4. Converting Between Data Types

```
double x = 9 / 2; // Should be 4.5
System.out.println(x); // prints 4.0

double y = 4;
System.out.println(y); //prints 4.0
```

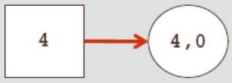
```
int    num1 = 7;
double num2 = 2;
double num3;
num3 = num1 / num2; // num3 is 3.5
```



- Automatic promotions:
  - If you assign a smaller type to a larger type:



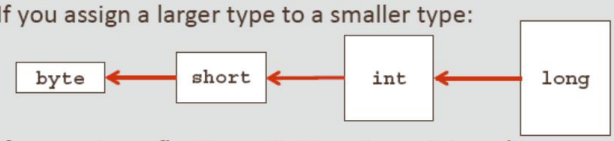

  - If you assign an integral value to a floating-point type:



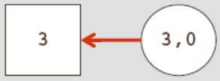

- Examples of automatic promotions:

```
-long intToLong = 6;
-double intToDouble = 4;
```

- When to cast:
  - If you assign a larger type to a smaller type:




  - If you assign a floating point type to an integral type:




- Examples of casting:

```
-int longToInt = (int)20L;
-short doubleToShort = (short)3.0;
```

double pi = 3.1416
int entero = (int) pi

127 in binary is 01111111; 128 in binary is 10000000.
Java uses the first bit in a number to indicates sign (+/-)

byte, short, and char values are automatically promoted to int prior to an operation

- Solution using larger data type:

```
int num1 = 53;
int num2 = 47;
int num3;
num3 = (num1 + num2);
```

Changed from byte to int

- Solution using casting:

```
int num1 = 53;              // 32 bits of memory to hold the value
int num2 = 47;              // 32 bits of memory to hold the value
byte num3;                  // 8 bits of memory reserved
num3 = (byte)(num1 + num2); // no data loss
```

## Automatic Promotion

- Example of a potential problem:

```
short a, b, c;
a = 1 ;
b = 2 ;
c = a + b ; //compiler error
```

a and b are automatically promoted to integers

- Example of potential solutions:
  - Declare c as an `int` type in the original declaration:
    - `int c;`
  - Type cast the (a+b) result in the assignment line:
    - `c = (short)(a+b);`

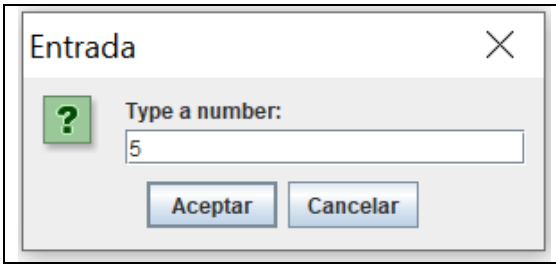
**int x = 123_456_789;**
**int x = 123456789;**

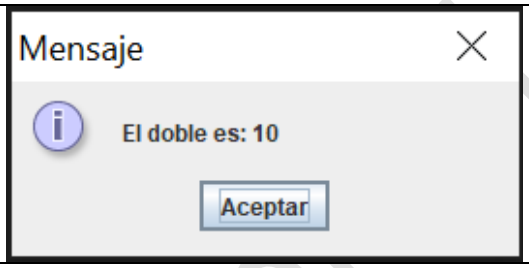
**intintVar1 = Integer.parseInt("100");**
**doubledoubleVar2 = Double.parseDouble("2.72");**

→

## 3.5. Keyboard Input

```
System.out.println("\033[H\033[2J"); // limpiar pantalla

String input = JOptionPane.showInputDialog(null, "Type a number:");
int number = Integer.parseInt(input);
number *= 2;
JOptionPane.showMessageDialog(null, "El doble es: " + number);
```




The Scanner searches for tokens

A few useful Scanner methods …

- nextInt() reads the next token as an int
- nextDouble() reads the next token as a double
- next() reads the next token as a String

```
Scanner sc = new Scanner(System.in);
```

**The Scanner class considers space as the default delimiter while reading the input**

Reading from a File

- nextLine() advances this Scanner past the current line and returns the input that was skipped
- findInLine("StringToFind") Attempts to find the next occurrence of a pattern constructed from the specified String, ignoring delimiters

```
Scanner sc = new Scanner(MyClase.class.getResourceAsStream("texto.txt"));
```

```
Scanner sc = new Scanner(System.in);
int  x = sc.nextInt();
double y = sc.nextDouble();
String z = sc.next();
String linea = sc.nextLine();
int numero = Integer.parseInt(z);
sc.close();
```

Quiz 1: JFo - Section 3 - L1-L2
Quiz 2: JFo - Section 3 - L3-L5

→

# 4. Java Methods and Library Classes

## 4.1. What Is a Method?

Instantiate an object

These classes outline objcts' …
Properties(fields)
Behaviors(methods)

### Variables for Objects

```
int          age  = 22;
String       str  = "Happy Birthday!";
Scanner      sc   = new Scanner();
Calculator   calc = new Calculator();
```

type — name — value

Method name
Method return type
Parameters

```
public double calculate(int x, double y){
   double quotient = x/y;
   return quotient;
}//end method calculate
```

Implementation

```
double tax = 0.05;
double tip = 0.15;

double person1 = 10;
double total1  = person1*(1 +tax +tip);
System.out.println(total1);

double person2 = 12;
double total2  = person2*(1 +tax +tip);
System.out.println(total2);
```

```
public void findTotal(double price, String name){
   double total = price * (1 + tax + tip);
   System.out.println(name + ": $ " + total);
} //end method findTotal
```

### Method Arguments and Parameters

- An argument is a value that's passed during a method call:

```
Calculator calc = new Calculator();
calc.calculate(3, 2.0);     //should print 1.5
```

Arguments

- A parameter is a variable that's defined in the method declaration:

```
public void calculate(int x, double y){
    System.out.println(x/y);
}//end method calculate
```

3 — 2.0 — Parameters

### Method Return Types

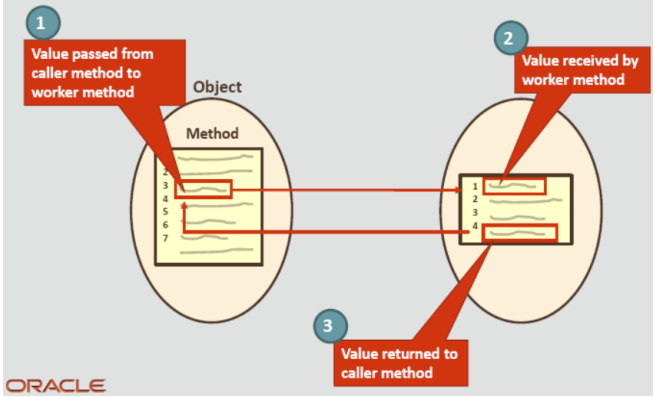
- Variables can have values of many different types:

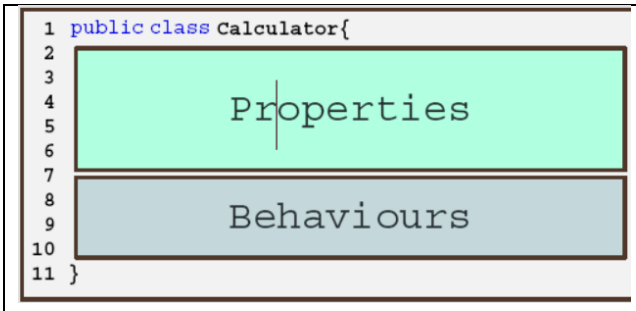
int, double, long, char, float, byte, short, String, boolean, Calculator

- Method calls also return values of many different types:

int, double, long, char, float, byte, short, String, boolean, Calculator

### Passing Arguments and Returning Values






```java
public class Calculator{
	//Fields
	public double tax = 0.05;
	public double tip = 0.15;
	public double originalPrice = 10;

	//Methods
	public void findTotal(){
	  //Calculate total after tax and tip
	  //Print this value
	}//end method findTotal

} //end class Calculator
```

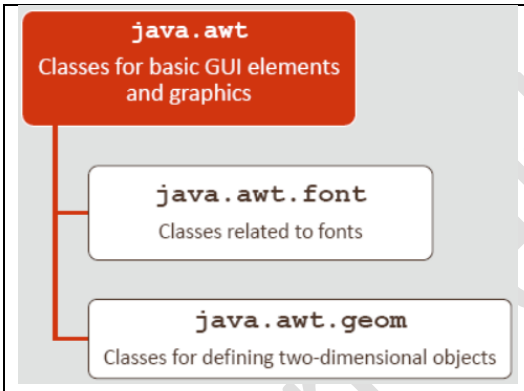
Calculator calc = **new** Calculator();

## 4.2. The import Declaration and Packages

java.base (Java SE 17 & JDK 17) https://docs.oracle.com/en/java/javase/17/docs/api/java.base/module-summary.html

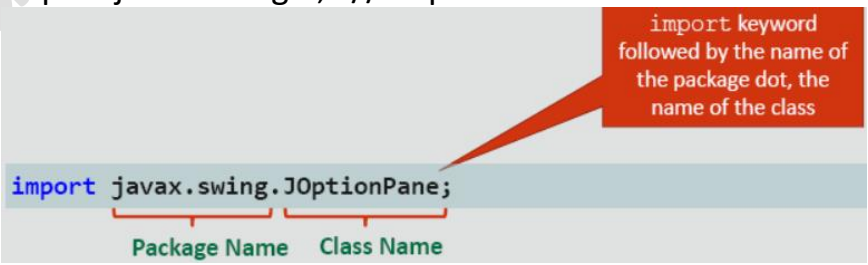
Overview (Java SE 15 & JDK 15) https://docs.oracle.com/en/java/javase/15/docs/api/index.html

| Package | Purpose |
|---|---|
| **java.lang** | Provides classes that are fundamental to the design of the Java language<br>By default, the java.lang package is automatically imported into all Java programs |
| **javax.swing** | Provides classes to build GUI components |
| **java.net** | Provides classes for networking applications |
| **java.time** | Providesclasses for dates, times, instants, and durations |




import


Import javax.swing.*; // importar todas las clases




**java.util.Scanner keyboard = new java.util.Scanner(System.in);**
**Scanner sc = new Scanner(System.in);**
**JOptionPane**.showMessageDialog(null, "Hello!");

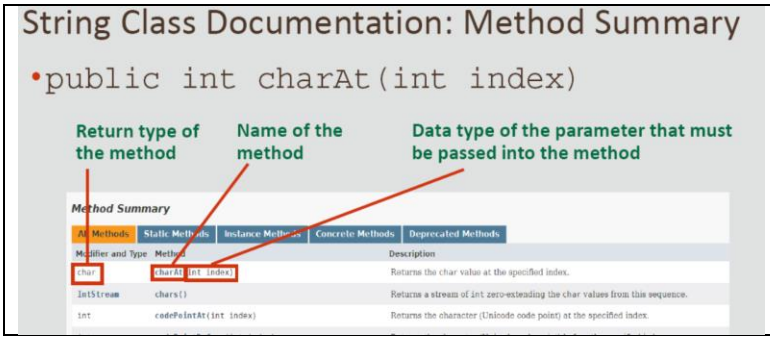
Quiz 1: JFo - Section 4 - L1-L2
Quiz 2: JFo - Section 4 - L3-L5

→

## 4.3. The String Class

java.lang.String
In Java, strings are not a primitive data type. Instead, they are objects of the String class.

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/module-summary.html
https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/lang/String.html



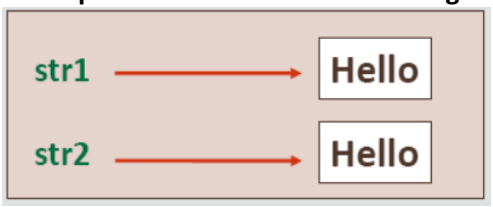

```
String str = "Hello, World";
```

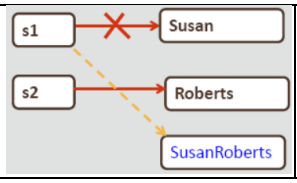
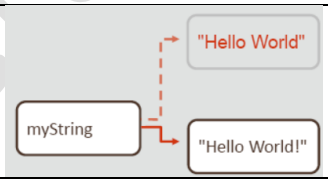
| H | e | l | l | o | , |  | W | o | r | l | d |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |

String str = new String("Hello, World");
Not commonly used and not recommended

| | |
|---|---|
| int length() | Returns the length of this string  Example:  LastName.length() |
| char charAt(int index) | Returns the char value at the specified index |
| String concat(String str) | Concatenates the specified string to the end of this string.<br>String producto = "coca";<br>producto.concat(" cola");  producto= producto.concat(" cola");<br>producto = producto +" cola"; |
| boolean contains(CharSequence s) | Returns true if and only if this string contains the specified sequence of char values. |
| int indexOf(String str) | Returns the index within this string of the first occurrence of the specified substring |
| int indexOf(char c) | Returns the index value of the first occurrence of c |
| int indexOf(char c, int beginIdx) | Returns the index value of the first occurrence of c,<br>starting from beginIdx to the end of the string |
| String substring(int beginIdx) | Returns the substring from beginIdx to the end of the string |
| String substring(in tbeginIdx, int endIdx) | Returns the substring from beginIdx up to,<br>but not including endIdx |
| String replace(char oldChar, char newChar)<br>String replace(CharSequence target,<br>CharSequence replacement) | This method replaces **all** occurrences of matching characters in a string |
| replaceFirst(String pattern, String replacement) | replaces only the first occurrence of a matching character pattern in a string |
| int lastIndexOf(String str)<br><br>int lastIndexOf(String str, int fromIndex)<br><br>String trim()<br><br>String toLowerCase()<br><br>String toUpperCase() | Investigar que hacen las siguientes funciones<br><br>cadena = "coca cola toma lo bueno"<br>Realizar el programa que regrese el número de palabras de cadena |

Strings Are Immutable, its value can't be changed.

| | |
|---|---|
| **String str1 = "Hello";**<br>**String str2 = "Hello";**<br>**We expect this . . . . . but it's wrong**<br>  | But this is what happens . . .<br>  <br>The Java runtime system knows that the two strings are identical and allocates the same memory location for the two objects. |

| | | | |
|---|---|---|---|
| String s1 ="Susan";<br>String s2 = "Roberts";<br>s1 = s1 + s2; |  | String myString = "Hello";<br>myString = myString.concat(" World");<br>myString = myString + "!" |  |

**Comparing String**

| | |
|---|---|
| **ASCII**<br>**'0' = 48**<br>**'1' = 49**<br>**'A' = 65**<br>**'a' = 97**<br><br>Practica: Imprimir el abecedario | // Conversiones explícitas<br>int ascii = (int) 'A';<br>char character = (char) ascii;<br><br>// Conversiones implícitas<br>int ascii1 = 65;<br>int ascii2 = 'A';<br>char caracter1 = 65;<br>char caracter2 = 'A'; |

The strings are compared character by character until their order is determined or until they prove to be identical
Syntax: **s1.compareTo(s2)** Example: int a = "computer".compareTo("comparison");
Returns an integer value that indicates the ordering of the two strings

- Returns == 0 when the two strings are lexicographically equivalent
- Returns < 0 when then the string calling the method is lexicographically first
- Returns > 0 when the parameter passed to the method is lexicographically first

→

## 4.4. The Random class

import java.util.Random;

- Random rand = new Random();
  **rand.setSeed(5L);** **Colocar una semilla**
- Math.random(); // entre 0 y 1

rand.nextInt(max - min + 1) + min;
(int) (Math.random() * (max - min + 1) ) + min;

| **Method** | **Produces** |
|---|---|
| boolean nextBoolean(); | A true or false value |
| int nextInt() | An integral value between Integer.MIN_VALUE and Integer.MAX_VALUE |
| long nextLong() | A long integral value between Long.MIN_VALUE and Long.MAX_VALUE |
| float nextFloat() | A decimal number between 0.0 (included) and 1.0 (excluded) |
| double nextDouble() | A decimal number between 0.0 (included) and 1.0 (excluded) |

→

## 4.5. The Math Class

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/module-summary.html

https://docs.oracle.com/en/java/javase/17/docs/api/java.base/java/math/package-summary.html

The methods of the Math class are **static methods**

Some of the Methods Available in Math Class

| **Method Name** | **Description** |
|---|---|
| abs(value) | absolute value |
| ceil(value) | rounds up |
| cos(value) | cosine, in radians |
| floor(value) | rounds down |
| log(value) | logarithm base e |
| log10(value) | logarithm base 10 |
| max(value1, value2) | larger of two values |
| min(value1, value2) | smaller of two values |
| pow(base, exponent) | base to the exponent power |
| random() | random double between 0 and 1 |
| round(value) | nearest whole number |
| sin(value) | sine, in radians |
| asin(value) | return radians |
| sqrt(value) | square root |

**double a = Math.sqrt(121.0);** **Math.E** **Math.PI**

**360º = 2π rad** **1º = π/180 rad** **1 rad = 180/π º**

**BMI = Peso en libras / Altura en pulgadas$^2$ * 703** **IMC = Peso (kg) ÷ (Altura (m))$^2$**

Sen(30º) = 0.5 arcsen(0.5) = 30º $sen^{-1}(0.5) = 30º$ asin(0.5) = 30º

→

# 5. Decision Statements

## 5.1. Boolean Expressions and if/else Constructs

In Java the values for the boolean data type are true and false, instead of yes and no.

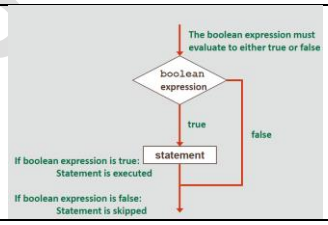
boolean bandera = true;
int x = 4;
boolean isFive = x == 5;

Relational Operators

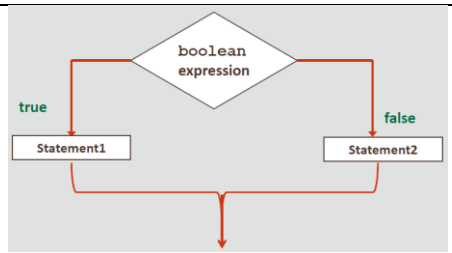
| Condition | Operator | Example |
|---|---|---|
| Is equal to | == | int i=1; (i == 1) |
| Is not equal to | != | int i=2; (i != 1) |
| Is less than | < | int i=0; (i < 1) |
| Is less than or equal to | <= | int i=1; (i <= 1) |
| Is greater than | > | int i=2; (i > 1) |
| Is greater than or equal to | >= | int i=1; (i >= 1) |

Conditional statements in Java are:
if statement
if/else statement
switch statement

```
if ( <some condition is true> ){

   //Statements will execute if the boolean
   //expression is true
}//endif
```




```
          boolean expression
if ( <some condition is true> ) {
       // do something
}                                  if block
else {
       // do something different   else block
}//endif
```




== compares the values of primitives
== compares the objects' locations in memory

| | |
|---|---|
| String x = "Ora";<br>String y = "cle";<br>String z = x +y;<br>boolean test = (z == x + y);<br>System.out.println(test); //false Why? | String x = "Ora";<br>String y = "cle";<br>String z = x +y;<br>boolean test = z.equals(x + y);<br>System.out.println(test); //true Why? |

## 5.2. Understanding Conditional Execution

Handling Multiple Conditions

```
int grade = 90;
int numberDaysAbsent = 0;

if (grade >= 88) {
    if (numberDaysAbsent == 0) {
        System.out.println("qualify");
    } // endif
} // endif
```

```
int grade = 90;
int numberDaysAbsent = 0;

if ((grade >= 88) && (numberDaysAbsent == 0)) {
   System.out.println("qualify");
} // endif
```

| Logic Operator | Meaning |
|---|---|
| && | AND |
\|\| | OR |
! | NOT |

```
boolean bandera = true;
if (bandera) {
    System.out.println("qualify");
} else {
    System.out.println("fail");
}
```

```
boolean bandera = true;
if (!bandera) {
    System.out.println("fail");
} else {
    System.out.println("qualify");
}
```

The && and || operators are short-circuit operators

Skipping the Second AND Test x=0 b = (x != 0) && ((y / x) > 2);

Skipping the Second OR Test x=0 b = (x <= 10) || (x > 20);

### Ternary Conditional Operator

| Operation | Operator | Example |
|---|---|---|
| If condition is true:<br>assign result = value1<br>Otherwise:<br>assign result = value2 | ?: | `result = condition ? value1 : value2`<br>Example:<br>`int x = 2, y = 5, z = 0;`<br>`z = (y < x) ? x : y;` |

```
int numberOfGoals = 1;
System.out.println("I scored " + numberOfGoals + " "
        + (numberOfGoals == 1 ? "goal" : "goals"));
```

```
if (<condition1>){
    //code_block1
} else if (<condition2>){
    // code_block2
} else if (<condition3>){
    // code_block3
} else {
    // default_code
} // endif
```

```
if (tvType == "color") {
   if (size == 14) {
      discPercent = 8;
   } else {
      discPercent = 10;
   }//endif
}//endif
. . . . . .
if (tvType == "color") {
   if (size == 14) {
      discPercent = 8;
   } //endif
} else {
   discPercent = 10;
}//endif
```

## 5.3. switch Statement

```
switch (<variable or expression>) {
    case <literal value>: //code_block1
                          [break;]
    case <literal value>: // code_block2
                          [break;]
    default: //default_code
 }//end switch
```

switch : variable  int, short, byte, char, or String

case : valor

break : Salir del switch

default : No encuentra relacion

### Solution: if/else Statement

```
Scanner in = new Scanner(System.in);
System.out.println("Enter your grade");
int grade = in.nextInt();
if (grade == 9){
   System.out.println("You are a freshman");
}
else if (grade == 10) {
   System.out.println("You are a sophomore");
}
else if (grade == 11) {
   System.out.println("You are a junior");
}
else if (grade == 12) {
   System.out.println("You are a senior");
}
else {
   System.out.println("Invalid grade");
}//endif
```

### Solution: switch Statement

```
Scanner in = new Scanner(System.in);
System.out.println("What grade are you in?");
int grade = in.nextInt();
switch (grade) {
   case 9:
       System.out.println("You are a freshman");
       break;
   case 10:
       System.out.println("You are a sophomore");
       break;
   case 11:
       System.out.println("You are a junior");
       break;
   case 12:
       System.out.println("You are a senior");
       break;
   default:
       System.out.println("Invalid grade");
}//end switch
```

**What Is switch Fall Through?**

- switch fall through is a condition that occurs if there are no break statements at the end of each case statement
- All statements after the matching case label are executed in sequence, regardless of the expression of subsequent case labels, until a break statement is encountered.

```
int month = 8;
month = in.nextInt();

switch (month) {
   case 1: case 3: case 5: case 7:
   case 8: case 10: case 12: System.out.print("31 days");
                             break;
   case 2: if(isLeapYear)){
            ..
```

Only a single value can be tested

Known values

→

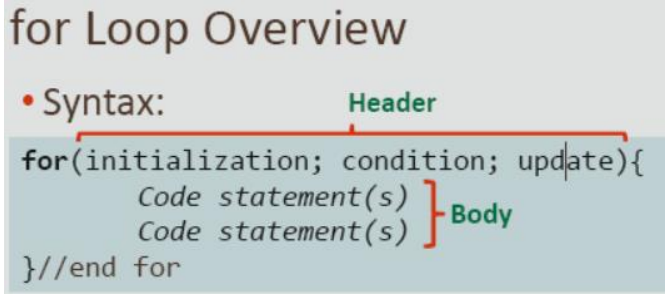
# 6. Loop Constructs

## 6.1. for Loops

El numero de ciclos o iteraciones es conocido

La inicialización de la variable solo se ejecuta la primera vez.

La ultima instruccion que se ejecuta **dentro** del ciclo es el incremento o decremento, posteriormente vuelve a iterar **mientras** se cumpla la condición.

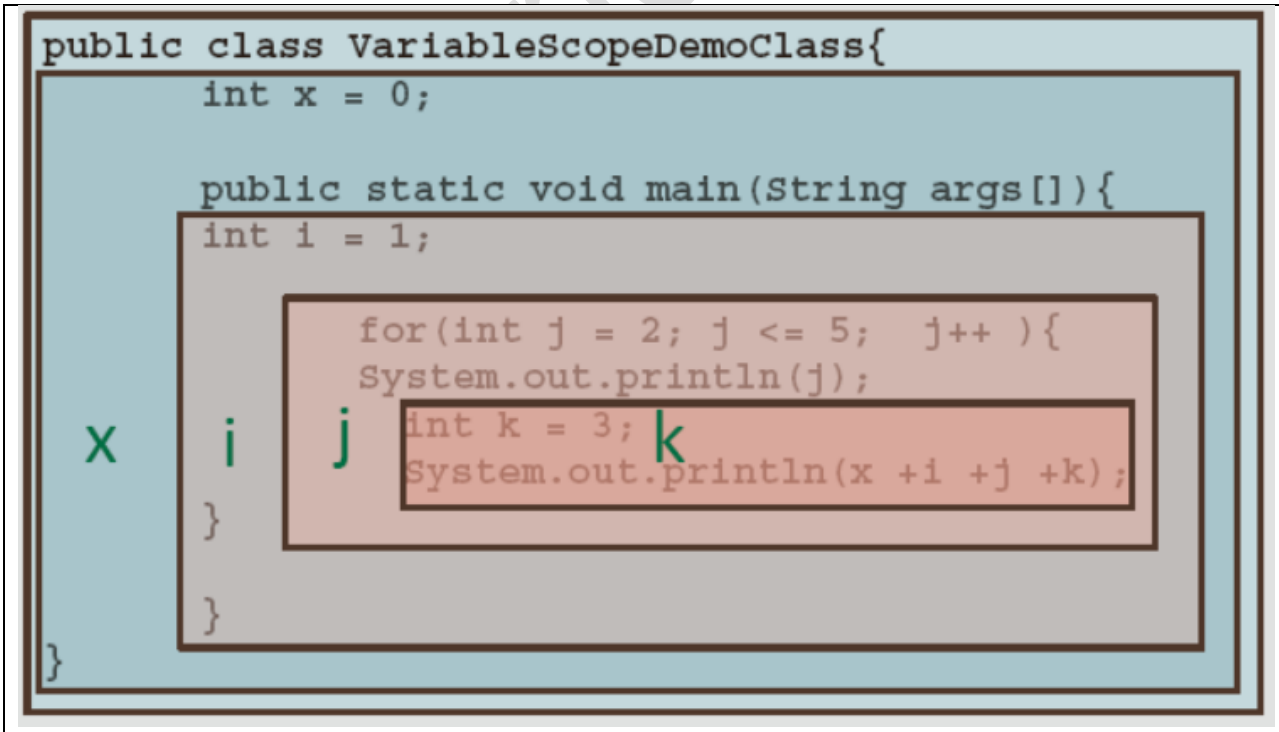



```
for ( ; ; ){
   System.out.println("Al infinito
                       y mas allá");
}
```

```
System.out.println("Countup to Song: ");
for (int i = 1;  i < 9;  i++) {
   System.out.println(i);
   // incremento implicito
} //end for
System.out.println("Mambo!");

System.out.println("Countdown to Launch: ");
int i;  // Scope
for (i = 10;  i >= 0;  i--) {
   System.out.println(i);
} //end for
System.out.println("Despegamos!: " + i );
```

**Variable Scope**

Variables cannot exist before or outside their block of code.



```
import java.util.Scanner;
public class PracticeCode {
public static void main(String[] args){
        Scanner in = new Scanner(System.in);
        int N = 100;
        int total = 0;
        System.out.println("This program adds " + N + " numbers.");
        for(int i = 0; i < N; i++){
        System.out.println(("Enter your next number:");
                int value = in.nextInt();
                total += value;
        }//end for
        System.out.println("The total is " + total + ".");
  }//end method main
```

N
total
i
value

Variable Already Defined

```
public static void main(String[] args) {
    int i = 0;
    for(int i = 64; i >0; i=i/2 ){
        System.out.print(i +" ");
    }
}
```

Out of Scope

```
public static void main(String[] args) {
    for(int j = 0; j<=5; j++){
        System.out.print(j +" ");
    }
    for(int j = 5; j>=0; j--){
        System.out.print(j +" ");
    }
    for(int k = 2; k<=64; k=k*2){
        System.out.print(j+" ");
    }
}
```

## 6.2. while and do-while loops

How Many Times to Repeat?

- In some situations, you don't know how many times to repeat something
- That is, you may need to repeat some code until a particular condition occurs

Pre-test

```
while (<boolean expression>) {
    <statement(s)> ;
}//end while
```

Post-test (mínimo una vez)

```
do{
 <statement(s)>
}while(<condition>);
```

The do-while loop requires a **semicolon** after the condition at the end of the loop

```
int i = 10;
System.out.println("Countdown to Launch!");
while (i >= 0) {
   System.out.println(i);
   i--; // i++;
} //end while
System.out.println("Blast Off!");
. . . . . .
int i = 10;
System.out.println("Countdown to Launch!");
do {
   System.out.println(i);
   i--;
} while (i >= 0);
System.out.println("Blast Off!");
```

Standard for Loop Compared with while Loop

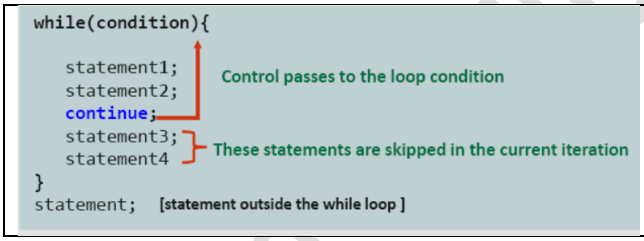
```
for (int i = 10; i >= 0; i--) {
   System.out.println(i);
}
System.out.println("Blast Off!");
```

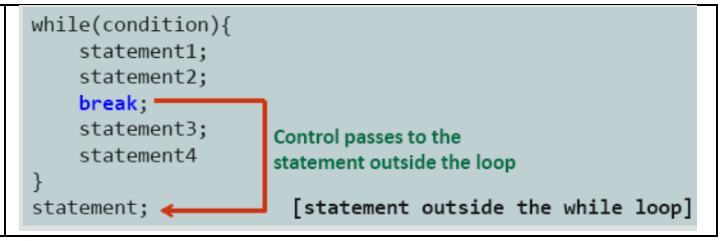
```
int i = 10;
while (i >= 0) {
   System.out.println(i);
   i--;
}
System.out.println("Blast Off!");
```

```
Scanner console = new Scanner(System.in);
int sum = 0;

System.out.println("Enter a number (-1 to quit): ");
int num = console.nextInt();
while (num != -1) {
    sum = sum + num;
    System.out.println("Enter a number (-1 to quit): ");
    num = console.nextInt();
} // end while
System.out.println("The sum is " + sum);
```

## 6.3. Using break and continue Statements

Use a **continue** statement to skip part of a loop up
Use a **break** statement to exit a loop down
Se pueden usar en cualquier ciclo: for, while, do while



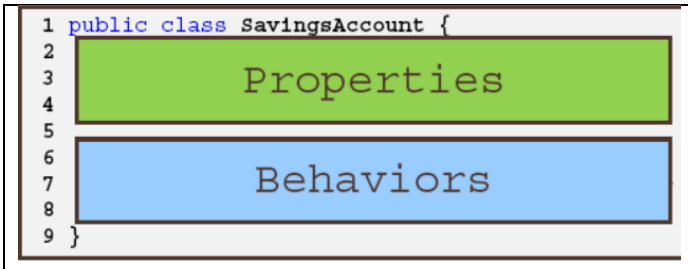

```
int i = 0;
while (i < 10) {
   if (i == 4) {
      break;
   }
   System.out.println(i+ "\t");
   i++;
}
System.out.println("\n. . .Fin");
```

→

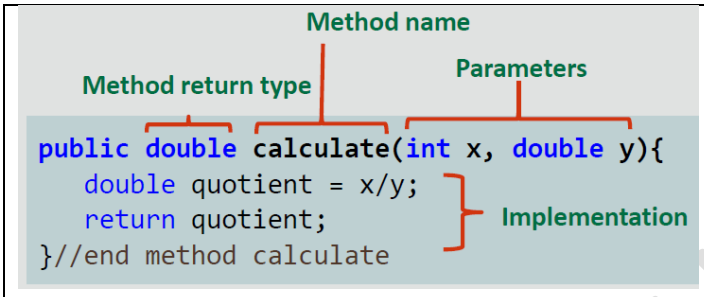
# 7. Creating Classes

## 7.1. Creating a Class

https://objectstorage.uk-london-1.oraclecloud.com/n/lrvrlgaqj8dd/b/Games/o/JavaPuzzleBall/index.html




```
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     }//end method displayCustomer
9 }//end class SavingsAccount
```




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%

En base a t(tiempo) obtener el rate(porcentaje)

```
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;
}
```

→

```java
public class Cuenta {
    int numeroID;  // Numero de Tarjeta
    String titular;
    double saldo;

    public double getSaldo() {
        return saldo;
    }

    public void depositar(double monto) {
        saldo += monto;
    }

    public void retirar(double monto) {
        if (monto <= saldo) {
            saldo = saldo - monto;
        } else {
            System.out.println(
                 "Sin suficiente saldo");
        }
    }
```

```java
public static void main(String[] args) {
    Cuenta cuenta1 = new Cuenta();
    cuenta1.numeroID = 1;
    cuenta1.titular = "Jesus";
    cuenta1.saldo = 1000;  // warning
    cuenta1.depositar(500);

    //Cuenta cuenta2 = new Cuenta(2, "Maria", 2000);
    //Cuenta cuenta3 = new Cuenta(3, "Jose",  3000);

    System.out.println(cuenta1.getSaldo());

    Cuenta[] cuentas = new Cuenta[3];
    cuentas[0] = cuenta1;
```

## 7.2. Instantiating Objects

Understand object references.
Understand the difference between stack and heap memory
Understand how Strings are special objects

```java
int x;
int y;

x = y;

x = 1;
y = 2;

System.out.println(x);
System.out.println(x == y);
```

¿Que imprime?

Strings Are Special Objects.

Strings should be instantiated without new
This is more memory-efficient

```java
String s1 = "Test";
```

But you shouldn't do this
String s2 = new String("Test");

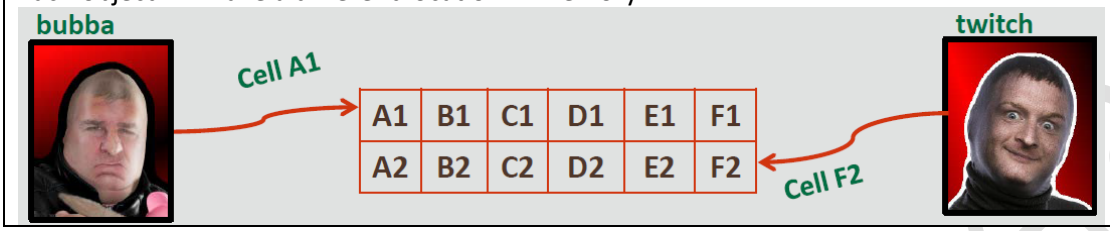
String s3 = "Test";

System.out.println(s1 == s2);  ??
System.out.println(s1 == s3);  ??

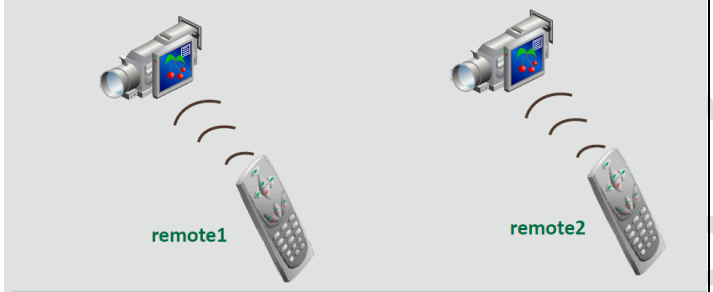
```java
public class Prisoner {
   public String name;
   public double height;
   public int sentence;
   LocalDate birthDate;
   LocalDate entryDate;
}
```

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


Each object will have a different location in memory



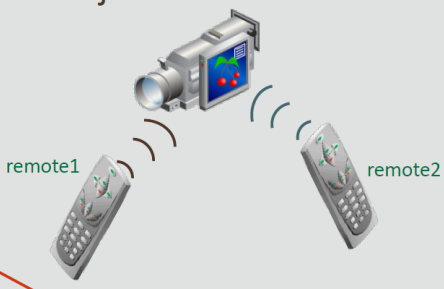

## Working with Object References: Example 1




```
Camera remote1 = new Camera();
Camera remote2 = new Camera();

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

There are two Camera objects

## Working with Object References: Example 2




```
Camera remote1 = new Camera();

Camera remote2 = remote1;

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

## 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();
```

## References to Different Objects: Example

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

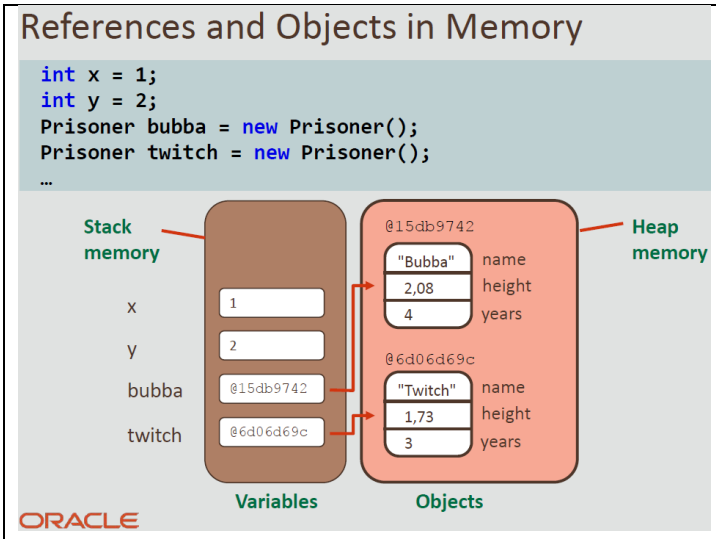
Caballo extends Animal { … } // El caballo **ES UN** Animal
Animal potro = new Caballo();

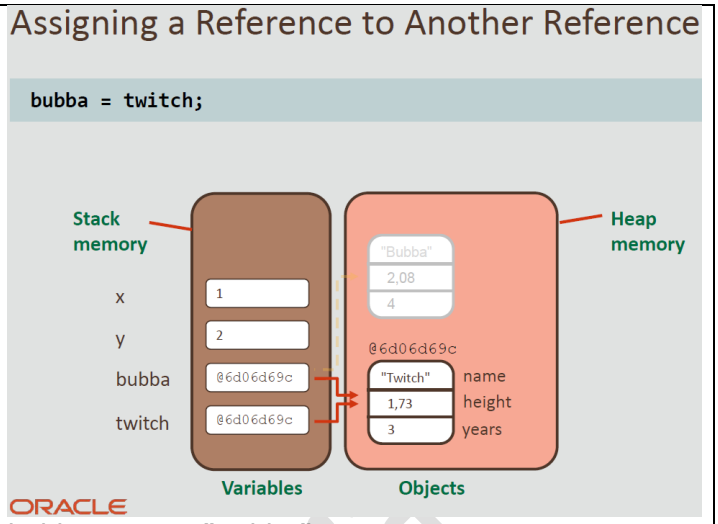
Stack memory is used to store …

- Local variables
- Primitives
- References to locations in the heap memory

Heap memory is used to store …

- Objects




```
bubba.name= "Bubba";
twitch.name= "Twitch";
System.out.println(bubba.name);        ??
System.out.println(twich.name);        ??
System.out.println(bubba == twitch);   ??
```

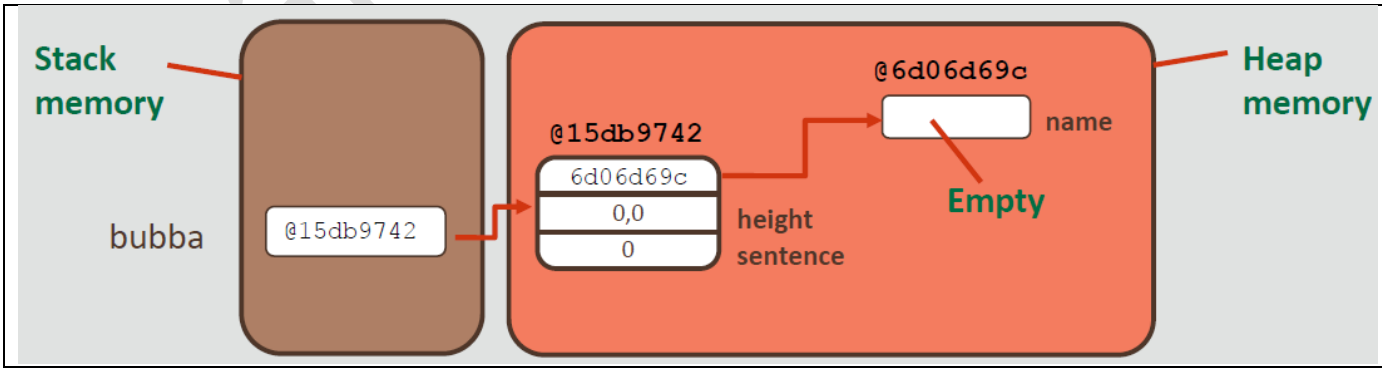
Si ninguna variable de referencia apunta a un objeto... Java borra automáticamente la memoria que ocupaba ese objeto. Esto se denomina recolección de basura (**Garbage Collection**). Los datos asociados a este objeto se pierden para siempre.

## 7.3. Constructors

| Data Type | Default Value |
|---|---|
| boolean | false |
| int | 0 |
| double | 0.0 |
| String | null |
| AnyObject type | null |

String test = null;
System.out.println(test.length()); // NullPointerException

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

   public static void main(String[] args) {
      Prisoner bubba  = new Prisoner(); //vacio
      System.out.println(bubba);
      bubba.name = "Bubba";
   }
}
```



| | |
|---|---|
| // Constructor<br>public Prisoner(String n, double h, int sentence) {<br>  name = n;<br>  height = h;<br>  this.sentence = sentence;<br>}<br>// They have no return type (not even void)<br>// They're named the same as the class | Prisoner bubba = new Prisoner("Bubba", 2.08, 4);<br>Prisoner twitch = new Prisoner("Twitch", 1.73, 3); |

Summary of Constructors

- Are special methods in a class
- Named the same as the class
- Have no return type (not even void)
- Called only once during object instantiation
- May accept arguments
- Used to set initial values of fields
- If you don't write your own constructor, Java provides a default zero-argument constructor



→