01 - Java Building Blocks - Exercices

1. Understanding the Java Class Structure

1.1. Escriba una clase con el modificador `public` y agregue otras clases en el mismo archivo sin modificador alguno.
- Retire el modificador `public` y constate que se mantiene compilando.
- Agregue el modificador `public` a las otras clases y verifique que ya no compila.

2. Writing a main() Method

2.1. Escribir una clase para cada una de las distintas formas del método `main()` debido a lista de parámetros y ejecutarlo para verificar su funcionamiento a nivel de línea de comandos.
- Compile a nivel de nivel de línea de comandos.
- Ejecute a nivel de nivel de línea de comandos: (1) llamando directo a la clase, usando el parámetro `-jar`

3. Understanding Package Declarations and Imports

3.1. Cree la clase `ImportExample` y retire los `imports` redundantes de:

```
import java.lang.System;
import java.lang.*;
import java.util.Random;
import java.util.*;
public class ImportExample {
  public static void main(String[] args) {
     Random r = new Random();
     System.out.println(r.nextInt(10));
  }
}
```

3.2. Cree la clase `Comflicts` y pruebe la ambigüedad de los siguientes `imports`. Prueba también utilizando `wildcards` para ver la preferencia del compilador:

```
import java.util.Date;
import java.sql.Date;

public class Conflicts {
  Date date;
  java.sql.Date sqlDate;
}
```

3.3. Cree los paquetes `com.ocajava8.pkga` para la clase `ClassA` y el paquete `com.ocajava8.pkgb` para la clase `ClassB`. El código para cada clase sería (agréguele las sentencias `package` e `imports` necesarios):

```
public class ClassA {
}

public class ClassB {
  public static void main(String[] args) {
    ClassA a;
    System.out.println("Got it");
```

```
   }
}
```

- Compile a nivel de línea de comandos y ejecute la clase `ClassB`

4. Creating Objects

4.1. En la siguiente clase se muestra bloques de inicialización:
- Cree una variable en el bloque 1 e intente utilizarla en el bloque 2. ¿Es posible?
- Asigne un valor a la variable `name` en el boque 2 y también el bloque 3. ¿con cuál valor finalmente se queda?
- Des comente el bloque 0, y corríjalo para que compila. ¿Cuál es el motivo de no compilar?

```
public class Blocks {
    //{ System.out.println("block 0:" + name); }
    private String name = "Manuel";
    { System.out.println("block 1"); }
    public Blocks() {
        name = "Pedro";
        System.out.println("setting constructor");
    }
    { System.out.println("block 2"); }
    public static void main(String[] args) {
        Blocks blocks = new Blocks();
        System.out.println(blocks.name);
    }
    { System.out.println("block 3"); }
}
```

5. Distinguishing Between Object References and Primitives

5.1 Trate de predecir la salida del siguiente código:

```
System.out.println(45);
System.out.println(0b10);
System.out.println(021);
System.out.println(0xA0);
```

5.2. Corrija el siguiente código para que compile, identificando el motivo:

```
double var1 = _120.00;
double var2 = 120.00_;
double var3 = 120_.00;
```

5.3. Cuál es el error en el siguiente código:

```
int vali = null;
String vals = null;
```

5.4. Intente usar la variable s1 en el siguiente código. En caso no compile, corrija el código para que compile:

```
String s1, s2;
String s3 = "yes", s4 = "no";
System.out.println("s1:" + s1);
```

5.4. Cuáles de los siguientes identificadores son legales/compilan/válidos. Des comente uno a uno para validar el resultado:

```
//int ok_niño;
```

```
//int $OK_deDía;
//int _alsoOK1d3ntifi3r;
//int __SStillOkbutKnotsonice$;
//int 3DPointClass;
//int hollywood@vine;
//int *$coffee;
//int private;
//int nombre_bajo;
```

6. Understanding Default Initialization of Variables

6.1. Cree la clase `DefaultValues` con atributos miembros sin inicializar (del tipo `String`, `int`, `double` y `boolean`) e inicialícelos para ver los valores por omisión, imprimiendo dichos valores.

6.2. Cree variables en un método de instancia y fíjese si se toma valores por omisión (es decir sin inicializar la variable).

6.3. Corrija el siguiente código para que compile:

```
public int notValid() {
    int y = 10;
    int x;
    int reply = x + y;
    return reply;
}
```

7. Understanding Variable Scope

7.1. Dado el siguiente código:
- Cuál sería la salida?
- ¿Si des comentamos el único código comentado, porque no compilaría?

```
public class Raton {
    static int MAX_LENGTH = 5;
    int length;
    public void crecer(int inches) {
        if (length < MAX_LENGTH) {
            int newSize = length + inches;
            length = newSize;
        }
    }
    public void comeSiTienesHambre(boolean tengoHambre) {
        if (tengoHambre) {
            int unPocoDeQueso = 1;
            {
                boolean unaMoridídita = true;
                System.out.println(unPocoDeQueso);
            }
        }
        //System.out.println(unaMoridídita);
    }
    public static void main(String[] args){
        Raton raton = new Raton();
        raton.crecer(4);
```

```
        System.out.println(raton.length);
        raton.crecer(3);
        System.out.println(raton.length);
    }
}
```

8. Ordering Elements in a Class

8.1. Para el siguiente código, cree la clase según el paquete mostrado y corríjalo para que compile (elimine lo que se requiera):
- Cuál es el problema con `//1`
- La variable creada en `//2` porque compila?, ¿qué tipo de variable es?

```
package pruebaordenclase;
import java.util.*;
String name; //1
public class PruebaOrden {
    double weight;
    public double getWeight() {
        return weight; }
    double height;   //2
}
```

9. Destroying Objects

9.1. Cree la clase `GCTest` con el código siguiente:
- Cuántos objetos son elegibles por el GC ejecutada la línea `//1`
- Cuántos objetos son elegibles por el GC ejecutada la línea `//2`
- Cuántos objetos son elegibles por el GC ejecutada la línea `//3`

```
public class GCTest {
    public static void main(String[] args) {
        String one, two;
        one = new String("a");
        two = new String("b");
        String three = one;
        one = two;
        one = null; //1
        two = three; //2
        two = null;
        three = two; //3
    }
 }
```

9.2. Cree una clase que incluya el método `finalize` y que muestre el mensaje `"Llamando a finalize..."`, y agregue un método `main` para probarlo. ¿Se ejecuta o no?, Por qué?

10. Benefits of Java

10.1. ¿Cuáles son algunos de los beneficios de programar en Java?