Taller 2

Juan David Balcazar Bedoya, código: 2879963

John Jairo Montoya Gómez, código: 2879540

1)

a) ¿Cuál es la importancia de la abstracción en el proceso de modelado?

La abstracción permite extraer las cualidades principales sin detenerse en los detalles, permitiendo generalizar, esto permite reducir el trabajo de implementación y aumentar la portabilidad del código

b) ¿Cuál es el código cliente (client code) y donde podemos encontrarlo?

Podemos encontrarlo para tener acceso a miembros de clases estáticas privadas de una clase sólo a través de métodos de esa clase. El código de cliente utiliza sólo superclases abstractas para reducir dependencias en específicos tipos de subclases. El código cliente también puede interactuar con clases genéricas.

c) ¿Cuáles son las similitudes y diferencias entre las variables de tipo primitivo (Primitive-type) y las variables de tipo referencia (reference-type)?

Las variables de tipo primitivo almacenan directamente un valor que siempre pertenece al rango de ese tipo.

Las variables de tipo referencia a objetos en cambio almacenan direcciones y no valores directamente. Una referencia a un objeto es la dirección de un área en memoria destinada a representar ese objeto.

d) Explique cuándo y por qué una clase debe proveer los métodos get/set para una variable de instancia.

Una clase debe proveer los métodos get/set cuando se quiere flexibilidad, extensibilidad y buen mantenimiento del código, para esto se utiliza el encapsulamiento, que nos permite proteger el código y así regular los valores que cada variable puede tener o haciendo que no se puedan acceder a ellas de forma directa.

e) El Garbage Collector elimina los objetos que ya no se utilizan proporcionando una gestión de memoria automática. ¿Cómo la JVM realiza el manejo de recursos? ¿Cuándo y cómo funciona el Garbage Collector?

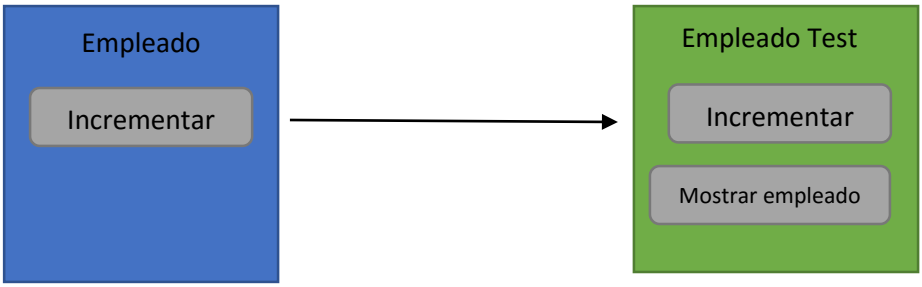
La memoria está dividida en tres partes, la zona de datos que es donde se almacenan las instrucciones del programa, las clases con sus métodos y constantes, Esta zona de

memoria es fija, y no se puede modificar durante el tiempo de ejecución. El Stack el cual el tamaño se define durante el tiempo de compilación y es estático durante su ejecución, cuando este se llena se genera un StackOverflow, Los datos que se almacenan aquí son las referencias a objetos. Y El Heap que es la zona de memoria dinámica, almacena los objetos que se crean, en un principio tiene un tamaño fijo asignado por la JVM pero según es necesario se va añadiendo más espacio.

El Garbage Collector es un proceso de baja prioridad que se ejecuta en la JVM y es el encargado de liberar la memoria que no se emplea. El ser de baja prioridad supone que no pueda estar todo el rato trabajando, y que solo se le asigne su tarea cuando el procesador no tiene un trabajo con mayor prioridad en ejecución.

f) Escriba, dibuje y codifique el ejemplo de delegación de este laboratorio.

Delegación es cuando básicamente se deja una tarea a otro objeto dentro de una clase. Aquí por ejemplo se crea un objeto Empleado dentro de EmpleadoTest pero realmente el trabajo lo hace el objeto de la clase Empleado . En el código del taller en el punto de Empleado hay delegación que es el método incrementar() que es delegado a el objeto empleado de la clase Empleado.




```
Class Empleado{

    void incrementar(double porcentaje){
    salarioMensual += salarioMensual * 0.01 * porcentaje;

}

}

Class EmpleadoTest {
```

```
  private empleado = new Empleado("Fulanito","Perez", 3251111);



  void incrementar(){


 empleado.incrementar(10); //Delegación
}
}
```

2) Rectángulo

```
package figura;

/**
 *
 * @author Juan David Balcázar Bedoya
 * Programación Orientada a Objetos 2015-2
 *  Taller 2 ejercicio 2
 *
 */
public class Rectangulo {

    private double largo;
    private double ancho;

    //Constructor por defecto
    public Rectangulo() {
        largo = 1;
        ancho = 1;
    }

    // constructor
    public Rectangulo(double largo, double ancho) {
        this.largo = largo;
        this.ancho = ancho;
    }

    // calcula perimetro
    public double calcularPerimetro(double height, double width) {
        double perimetro;

        if ((height <= 0.0) || (height > 20.0)          //revisa que el alto y
```

```
ancho este entre 0 y 20
                || (width <= 0.0) || (width > 20.0)) {
            System.out.println("No se puede calcular perimetro");
            return 0;
        } else {
            perimetro = (2 * height) + (2 * width);
            return perimetro;
        }

    }

    public double calcularArea(double height, double width) {
        double area;

        if ((height <= 0.0) || (height > 20.0)            //revisa que el alto y
ancho este entre 0 y 20
                || (width <= 0.0) || (width > 20.0)) {
            System.out.println("No se puede calcular area");
            return 0;
        } else {
            area = height * width;
            return area;
        }
    }

    public double getLargo() {
        return largo;
    }

    public void setLargo(double largo) {
        if ((largo <= 0.0) || (largo > 20.0)) {
           // System.out.println("Numero no permitido");
        } else {
            this.largo = largo;
        }

    }

    public double getAncho() {
        return ancho;
    }

    public void setAncho(double ancho) {
        if ((ancho <= 0.0) || (ancho > 20.0)) {
            //System.out.println("Numero no permitido");
        } else {
            this.ancho = ancho;
        }
    }
```

```
}


package figura;

import java.util.Random;

/**
 *
 * @author Juan David Balcazar Bedoya
 * Programación Orientada a Objetos 2015-2
 *  Taller 2 ejercicio 2
 *
 */
public class Main {

    public static void main(String[] args) {

        Rectangulo rect = new Rectangulo();
        double height = rect.getLargo();
        double width = rect.getAncho();
        System.out.println("Rectangulo 1 (por defecto)");
        System.out.println("Area: " + rect.calcularArea(height, width));
        System.out.println("Perimetro: " + rect.calcularArea(height, width) +
"\n");

        Random r = new Random();
        for (int i = 1; i <= 9; i++) {
            // min y max para el rango
            float min = -1.0f;
            float max = 25.0f;

            // random de largo y ancho entre 0 y 100
            float largo = r.nextFloat() * (max - min) + min;
            float ancho = r.nextFloat() * (max - min) + min;

            Rectangulo rect2 = new Rectangulo(largo, ancho);
            rect2.setAncho(ancho);
            rect2.setLargo(largo);

            //muestra datos de ancho, alto y calcula area y perimetro
            System.out.println("Rectangulo " + (i + 1));
            System.out.println("largo: " + rect2.getLargo());
            System.out.println("ancho: " + rect2.getAncho());
            System.out.println("Area: " + rect2.calcularArea(rect2.getLargo(),
rect2.getAncho()));
            System.out.println("Perimetro: " +
rect2.calcularPerimetro(rect2.getLargo(), rect2.getAncho()) + "\n");
        }
    }
```

```
}
```

3) Clase empleado

```
public class Empleado {

    private String nombre = null;
    private String apellido = null;
    private double salarioMensual;
    //Constructor que inicializa las variables de instancia
    public Empleado(String nombre, String appellido, double salarioMensual){

       this.nombre = nombre;
       this.apellido = appellido;
       //para asegurar que no es menor que zero
       if(salarioMensual < 0)
              this.salarioMensual = 0;
       else
           this.salarioMensual = salarioMensual;
    }
    //incrementa al porcentaje dado
    public void incrementar(double porcentaje){

       salarioMensual += salarioMensual * 0.01 * porcentaje;
    }
    //getters para las tres variables
    public String getNombre(){

       return nombre;
    }
    public String getApellido(){

       return apellido;
    }
    public double getSalarioMensual(){

       return salarioMensual;
    }
}
```

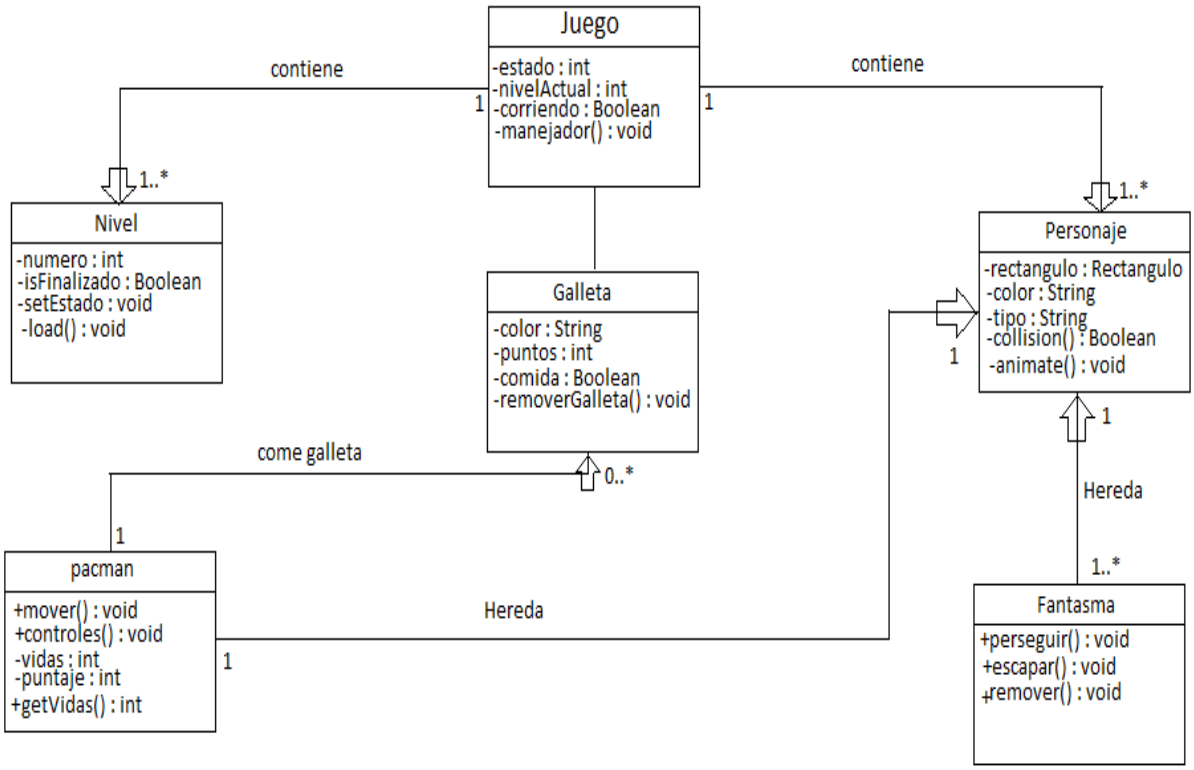
```
public class EmpleadoTest extends Thread {

       Empleado empleado1;
       Empleado empleado2;
       Empleado empleado3;
       Empleado empleado4;
       Empleado empleado5;
       //Constructor para inicializar los cinco empleados
```

```
	public EmpleadoTest(){

		 empleado1 = new Empleado("Juan", "Perez", 664350);
		 empleado2 = new Empleado("Mike", "Peralta",780000);
		 empleado3 = new Empleado("Dominick", "Smith", 1200000);
		 empleado4 = new Empleado("Carl", "White", 3500000);
		 empleado5 = new Empleado("Tom", "Holman", 5000000);
	}

	public void run(){

		mostrarEmpleados();
		incrementar();
	}
	//Muestra nombre y salario de los empleados
	private void mostrarEmpleados(){

		System.out.println(""
				+ empleado1.getNombre() + " " + empleado1.getApellido()
				+ "- Salario: " + empleado1.getSalarioMensual() + "\n"
				+ empleado2.getNombre() + " " + empleado2.getApellido()
				+ "- Salario: " + empleado2.getSalarioMensual() + "\n"
				+ empleado3.getNombre() + " " + empleado3.getApellido()
				+ "- Salario: " + empleado3.getSalarioMensual() + "\n"
				+ empleado4.getNombre() + " " + empleado4.getApellido()
				+ "- Salario: " + empleado4.getSalarioMensual() + "\n"
				+ empleado5.getNombre() + " " + empleado5.getApellido()
				+  "-  Salario:  "  +  empleado5.getSalarioMensual()  +
"\n");
	}
	//Incrementa llamando el metodo incrementar() de cada
	//empleado.
	public void incrementar(){

	    System.out.println("Salario inicial" + "\n");
	    mostrarEmpleados();

	    try {
			Thread.sleep(1500);
	      } catch (InterruptedException e) {
			e.printStackTrace();
	      }

	    int years = 0;
	    //Muestra el salario despued de cada 5 anos y espera
	    //segundo y medio para mostrar los siguientes.
	    while(years < 70){

	      years += 5;
	      System.out.println("En " + years + " anos los salarios seran:" +
"\n");
	      empleado1.incrementar(10); //incrementar al 10%
	      empleado2.incrementar(10);
	      empleado3.incrementar(10);
```

```
            empleado4.incrementar(10);
            empleado5.incrementar(10);
            mostrarEmpleados();

            try {
                        Thread.sleep(1500);
                  } catch (InterruptedException e) {
                        e.printStackTrace();
                  }

           }

      }

}

public class Taller2 {

      public static void main(String[] args) {

            //Comenzar hilo test
          EmpleadoTest et = new EmpleadoTest();
          et.start();

      }

}
```

La clase **EmpleadoTest** usa la clase Empleado para desplegar los datos. Primero se crean cinco instancias de empleado con sus respectivos nombres, apellidos y salario mensual. El método **incrementar** de la clase empleado nos adiciona el 10% del salario. En el **while** de la clase **EmpleadoTest** donde se llama el método **incrementar** en cada ciclo hasta que los años sean menores que 70, aumentando de cinco en cinco.

4)Pac-man



La clase Juego contiene las clases Nivel, Personaje, Galleta, Pacman y Fantasma. Pacman y Fantasma heredan de la clase Personaje. Pacman puede comer galletas que son clases que forman objetos que desaparecen al removerlos con el método removeGalleta(). Pacman tiene tamaño dado por el rectángulo y se puede mover usando el método mover() y controles(). La clase Fantasma tiene dos métodos que le permiten perseguir o escapar de pacman. El juego presenta diferentes estados como optiones, nivel etc para lo que se una un Enum. Los detalles gráficos del nivel son cargados con el método load();

Bibliografía

http://users.dcc.uchile.cl/~lmateu/Java/Apuntes/tiposprim.htm

http://pages.cs.wisc.edu/~bahls/cs302/PrimitiveVsReference.html

https://ayddup.wordpress.com/2011/08/08/la-memoria-en-java-garbage-collector-y-el-metodo-finalize/

H.M. Deitel and P.J. Deitel, Java How to Program)