Universidad Nacional de Colombia. Sede Bogotá
Facultad de Ingeniería
Departamento de Ing. Industrial y Sistemas
Castro Castro Sergio, Isea Noguera Raúl, Ortiz Ballén Jessica

Códigos utilizados para la implementación de cada uno de los ejercicios (incluyen comentarios):

Ejercicio 1: Responda las siguientes preguntas

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

La importancia de la abstracción radica en que podemos identificar las características y funcionalidades esenciales o primarias que hacen de un objeto, lo que es. Y como la programación orientada a objetos trata de crear esos objetos de una manera abstracta, al identificar esas características nos da una plantilla de ese objeto, facilitando el manejo o solucionando problemas que involucren a ese objeto.

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

El código cliente es la base del programa que cono la ip o nombre del dominio o hostname del programa servidor y el puerto que está escuchando, y solicita el establecimiento de conexión con el servidor, al conectase es cuando intercambian información. Esto se denomina Sockets en Java o interconexión entre Cliente-Servidor. (Client code) Para estabilizar un servidor entonces se crea un servidor socket, luego el servidor puede utilizar la siguiente declaración para escuchar las conexiones:

Socket Socket = serverSocket.accept ();

Esta declaración de espera hasta que un cliente se conecta a la toma de servidor. El cliente emite la siguientedeclaración para solicitar una conexión a un servidor:

Socket socket = new Socket (serverName, puerto);

Esta declaración abre un socket de modo que el programa cliente puede comunicarse con el servidor.serverName es el nombre de host de Internet del servidor o la dirección IP. La siguiente sentencia crea un socket en el puerto 8000 en la máquina cliente para conectarse al 130.254.204.36 host:

Socket socket = new Socket ("130.254.204.36", 8000)

Alternativamente, puede utilizar el nombre de dominio para crear un socket, de la siguiente manera:

Socket socket = new Socket ("drake.armstrong.edu", 8000);

Cuando se crea un socket con un nombre de host, la JVM pide al DNS para reducir el nombre de host en la dirección IP.

También un programa puede utilizar el nombre de host local host o la dirección IP 127.0.0.1 para referirse a la máquina en la que se está ejecutando un cliente, así como el constructor Socket lanza una java.net.UnknownHostException si el host no puede ser encontrado.

Ahora el Client. Java es:

```
1 import java.io.*;
2 import java.net.*;
3 import java.awt.*;
4 import java.awt.event.*;
5 import javax.swing.*;
6
7 public class Client extends JFrame {
8 // Text field for receiving radius
9 private JTextField jtf = new JTextField();
10
11 // Text area to display contents
12 private JTextArea jta = new JTextArea();
13
14 // IO streams
15 private DataOutputStream toServer;
16 private DataInputStream fromServer;
17
18 public static void main(String[] args) {
19 new Client();
20 }
21
22 public Client() {
23 // Panel p to hold the label and text field
24 JPanel p = new JPanel();
25 p.setLayout(new BorderLayout());
26 p.add(new JLabel("Enter radius"), BorderLayout.WEST);
27 p.add(jtf, BorderLayout.CENTER);
28 jtf.setHorizontalAlignment(JTextField.RIGHT);
29
30 setLayout(new BorderLayout());
31 add(p, BorderLayout.NORTH);
32 add(new JScrollPane(jta), BorderLayout.CENTER);
33
34 jtf.addActionListener(new TextFieldListener());
```

```
35
36 setTitle("Client");
37 setSize(500, 300);
38 setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
39 setVisible(true); // It is necessary to show the frame here!
40
41 try {
42 // Create a socket to connect to the server
43
44 // Socket socket = new Socket("130.254.204.36", 8000);
45 // Socket socket = new Socket("drake.Armstrong.edu", 8000);
46
47 // Create an input stream to receive data from the server
48
49
50
51 // Create an output stream to send data to the server
52
53
54 }
55 catch (IOException ex) {
56 jta.append(ex.toString() + '\n');
57 }
58 }
59
60 private class TextFieldListener implements ActionListener {
61 public void actionPerformed(ActionEvent e) {
62 try {
63 // Get the radius from the text field
64 double radius = Double.parseDouble(jtf.getText().trim());
65
66 // Send the radius to the server
67
68
69
70 // Get area from the server
71 double area = ;
72
73 // Display to the text area
74 jta.append("Radius is " + radius + "\n");
75 jta.append("Area received from the server is "
76 + area + '\n');
77 }
78 catch (IOException ex) {
79 System.err.println(ex);
80 }
81 }
```

82 }
83 }

c. ¿Cuáles son las similitudes y diferencias entre las variables de tipo primitivo (Primitive-Type) y las variables de tipo referencia (Reference-Type)?

Para empezar, se define que es una variable. Una variable es una posición en memoria reservada para almacenar valores, de ellos hay 2 tipos:

- *Tipos de datos Primitivos: Las variables de este tipo de datos, almacena un valor que pertenece al rango de ese tipo y java crea ese valor automáticamente (no como crear con la palabra new) para que sea más eficiente. Los tipos de variables primitivos son:*
  - *boolean*
  - *char*
  - *byte*
  - *short*
  - *int*
  - *long*
  - *float*
  - *double*
  - *void*
- *Tipos de datos de Referencia: Estos tipos de datos se crean mediante constructores, se utilizan para acceder a los objetos además de almacenar valores; y a diferencia de los tipos primitivos, estos permiten realizar acciones más especificas con las variables creadas, estos tipos de variables son:*
  - *Boolean*
  - *Character*
  - *Byte*
  - *Short*
  - *Integer*
  - *Long*
  - *Float*
  - *Double*
  - *Void*

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

Las variables de instancia son aquellas que se definen dentro de una clase como públicas y es allí donde no se requieren los métodos para tomar o modificar su valor, lo cual altera el control de la clase sobre la misma variable, lo cual no es recomendable, por ello se definen como privadas y así no se puede modificar o obtener desde otra clase, solo a través de los métodos get y set pueden ser modificadas u obtenidas; estos métodos provienen de una clase y son utilizados por objetos dentro de otra clase. Un Get es un método que obtiene el valor de una propiedad específica, y el Set es un método que establece el valor de una propiedad específica, estos se definen desde cualquier objeto predefinido en el núcleo o de un objeto definido por el usuario que admita la adición de nuevas características, utilizando la sintaxis literal de un objeto.

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 JVM no sabe del lenguaje de programación Java, solo en formato binario particular, el formato archivo de clase. Un archivo de clase Java contiene instrucciones de máquina virtual (o códigos de bytes) y una tabla de símbolos, así como otra información auxiliar.*

  En aras de la seguridad, la máquina virtual Java impone fuertes restricciones sintácticas y estructurales en el código en un archivo de clase. Sin embargo, cualquier idioma con la funcionalidad que puede ser expresado en términos de un archivo de clase válida puede ser recibido por la máquina virtual Java. Atraídos por una plataforma generalmente disponibles, independiente de la máquina, los ejecutores de otros idiomas puede dar vuelta a la máquina virtual de Java como un vehículo de entrega de sus lenguajes.
  Hay tres componentes de la JVM que se centran en sintonizar el rendimiento. El montón es donde se almacenan los datos objeto. Esta área es administrada entonces por el recolector de basura seleccionada en el arranque. La mayoría de las opciones de ajuste se refieren a dimensionar el montón y eligiendo el recolector de basura más apropiado para su situación. El compilador JIT también tiene un gran impacto en el rendimiento.

- *El GC es un recolector de basura por elección para la mayoría de las aplicaciones que no tienen requisitos de bajo tiempo de pausa, y se ejecutan en máquinas tipo cliente. Se aprovechan de un solo procesador virtual para la recolección de basura. Por ello en el hadware actual, la GC puede gestionar eficientemente una gran cantidad de aplicaciones no triviales con unos pocos cientos de MB de almacenamiento dinámico de Java, con relativamente pocas pausas. Otro uso es para loa entornos donde un alto número de JVMs se ejecutan en la misma máquina, más que los procesadores disponibles, en este caso cuando de GC hace la recolección de*

basura, es mejor usar un solo procesador para minimizar la interferencia en las JVM restantes, incluso si la recolección de basura pueda durar más tiempo, lo cual se ajusta al GC.

- *El primer proceso se denomina marcado, este es cuando el GC identifica cuales son las piezas de memoria que son usadas y cuáles no. Objetos referenciados se muestran en azul, los no referenciados en dorado. Todos los objetos son escaneados y analizados en este proceso, lo cual hace que sea dispendioso.*
- *Borrado normal, elimina objetos no referenciados dejando objetos referenciados y punteros para liberar espacio. El asignador de memoria contiene referencias a bloques de espacio libre donde el objeto puede ser asignado.*
- *Borrado de compactación, además de la eliminación de elementos no referenciados, también se puede compactar los objetos referenciados restantes. Moviendo el objeto referenciado en conjunto, esto hace que la nueva asignación en la memoria sea mucho más fácil y rápida.*

Ejercicio 2: Rectángulo:

a. Cree una clase rectángulo. La clase tiene atributos largo y ancho, cada una con un valor por defecto de 1. Esta debe tener los métodos que calculen el perímetro y el área del rectángulo. Para sus métodos de acceso, las funciones ser deben verificar que el largo y ancho sea una variable de tipo real, que no sea menor a 0.0 ni mayor a 20.0.

```
public class Rectangulo {
    double largo= 1.0;
    double ancho= 1.0;

    Rectangulo (){
        }
    Rectangulo (double newLargo,double newAncho){
        largo=newLargo;
        ancho=newAncho;
    }
    double setPerimetro(){
        if (0.0< largo && largo<=20.0 && 0.0<ancho && ancho<=20.0){
        return (largo+ancho)*2;
        }
        else{
            return 0;
        }
    }
    double setArea(){
        if (0.0< largo && largo<=20.0 && 0.0<ancho && ancho<=20.0){
        return largo*ancho;
        }
        else{
            return 0;
```

```
        }
    }
}
```

b. Escriba una Clase Main donde se creen 10 nuevos rectángulos con diferentes características (debe haber uno creado con valores por defecto), consulte e imprima a todos por el área y perímetro.

```
import java.util.Scanner;


public class pruebarectangulo {


   public static void main(String[] args) {

      Rectangulop rectangulo1= new Rectangulop( 10,20 );

      System.out.println("El perimetro del rectangulo de largo: "+rectangulo1.largo+" y ancho "+rectangulo1.ancho+" es: "+rectangulo1.setPerimetro());

      System.out.println("El area del rectangulo de largo: "+rectangulo1.largo+" y ancho "+rectangulo1.ancho+" es: "+rectangulo1.setArea());


     int i=1;

     while(i<10){

      double a,b;

      System.out.println("Ingrese el largo del rectangulo");

         Scanner sc = new Scanner(System.in);

         a= sc.nextInt();


      System.out.println("Ingrese el ancho del rectangulo");

         Scanner si = new Scanner(System.in);

         b= sc.nextInt();


      Rectangulop rectangulo2= new Rectangulop( a,b );

      System.out.println("El perimetro del rectangulo de largo: "+rectangulo2.largo+" y ancho "+rectangulo2.ancho+" es: "+rectangulo2.setPerimetro());

      System.out.println("El area del rectangulo de largo: "+rectangulo2.largo+" y ancho "+rectangulo2.ancho+" es: "+rectangulo2.setArea());
```

```
        i++;
       }
     }
   }
    class Rectangulop {
    double largo= 1.0;
    double ancho= 1.0;

    Rectangulop (double newLargo,double newAncho){
       largo=newLargo;
       ancho=newAncho;
    }
    double setPerimetro(){
       if (0.0< largo && largo<=20.0 && 0.0<ancho && ancho<=20.0){
       return ((largo+ancho)*2);
       }
       else{
          return 0;
       }
    }
    double setArea(){
       if (0.0< largo && largo<=20.0 && 0.0<ancho && ancho<=20.0){
       return (largo*ancho);
       }
       else{
          return 0;
       }
    }
}
```

Ejercicio 3: Empleados:

a. Cree una clase llamada “Empleado” que incluya 3 atributos, estos pueden ser la variables de instancia como nombre (tipo String), apellido(tipo String) y salario mensual (tipo double). Tu clase debe tener un constructor que inicialice los 3 atributos,

también debe tener los métodos de acceso para cada variable de instancia. Debe validar que el salario mensual nunca sea negativo, en ese caso asígnelo como 0.0

```
public class Empleado {
	String name;
	String apellido;
	double salario_mensual;

	public Empleado(String name, String apellido, double salario_mensual) {
		this.name = name;
		this.apellido = apellido;
		this.salario_mensual = salario_mensual;
	}

	public String getName() {
		return name;
	}

	public void setName(String name) {
		this.name = name;
	}

	public String getApellido() {
		return apellido;
	}

	public void setApellido(String apellido) {
		this.apellido = apellido;
	}

	public double getSalario_mensual() {
		return salario_mensual;
	}

	public void setSalario_mensual(double salario_mensual) {
		if(salario_mensual < 0)
			this.salario_mensual = 0.0;
		else
			this.salario_mensual = salario_mensual;
	}


	public double getSalarioAnual() {
		return salario_mensual * 12;
	}

	public double getSalarioAumento() {
		salario_mensual = salario_mensual + ((salario_mensual * 10)/100);
		return salario_mensual * 12;
	}
```

}

b. Escriba una clases que realice la prueba de la clase “Empleado” llamada “EmpleadoTest” la cual demuestre las capacidades que tiene la clase “Empleado”. Cree 5 objetos empleados los cuales muestren su salario anual.

Luego incremente por 5 años el salario mensual de los empleados en un 10% y por cada incremento imprima el salario anual.

Ejemplo para 2 empleados:

Empleado 1: Jack Pitman; Salario Anual 12300.00
Empleado 2: Gilliam Wyatt; Salario Anual 5432.00
...Incremento del 10%

Empleado 1: Jack Pitman; Salario Anual 13530.00
Empleado 2: Gilliam Wyatt; Salario Anual 5975.20
...

```
import java.util.*;

public class EmpleadoTest {
        public static void main(String[] args) {

                Scanner sc = new Scanner(System.in);

                Empleado empleado1 = new Empleado("Jhon", "Salas", 45500);
                Empleado empleado2 = new Empleado("Luis", "Boling", 23654);
                Empleado empleado3 = new Empleado("Jane", "Muse", 6840);
                Empleado empleado4 = new Empleado("Esther", "Cameron", 84631);
                Empleado empleado5 = new Empleado("Charles", "Cousteau", 9846);

                System.out.println("Empleado 1: " + empleado1.name + " " +
empleado1.apellido + "; Salario Anual: " + empleado1.getSalarioAnual());
                System.out.println("Empleado 2: " +empleado2.name + " " +
empleado2.apellido + "; Salario Anual: " + empleado2.getSalarioAnual());
                System.out.println("Empleado 3: " +empleado3.name + " " +
empleado3.apellido + "; Salario Anual: " + empleado3.getSalarioAnual());
                System.out.println("Empleado 4: " +empleado4.name + " " +
empleado4.apellido + "; Salario Anual: " + empleado4.getSalarioAnual());
                System.out.println("Empleado 5: " +empleado5.name + " " +
empleado5.apellido + "; Salario Anual: " + empleado5.getSalarioAnual());
                System.out.println("");
                System.out.println("Presione 5 para saber el aumento salarial cada 5
años");
                int x;
                while(sc.hasNext()){
                        x = sc.nextInt();
                        if(x == 5){
                                System.out.println("El salario mensual ha aumentado
un 10%. Los salarios anuales son los siguientes");
```

```
                                        System.out.println("Empleado 1: " +empleado1.name +
" " + empleado1.apellido + "; Salario Anual: " + empleado1.getSalarioAumento());
                                        System.out.println("Empleado 2: " +empleado2.name +
" " + empleado2.apellido + "; Salario Anual: " + empleado2.getSalarioAumento());
                                        System.out.println("Empleado 3: " +empleado3.name +
" " + empleado3.apellido + "; Salario Anual: " + empleado3.getSalarioAumento());
                                        System.out.println("Empleado 4: " +empleado4.name +
" " + empleado4.apellido + "; Salario Anual: " + empleado4.getSalarioAumento());
                                        System.out.println("Empleado 5: " +empleado5.name +
" " + empleado5.apellido + "; Salario Anual: " + empleado5.getSalarioAumento());
                                        System.out.println("");
                                }else{
                                        System.out.println("Por favor presione el numero 5.");
                                }
                        }

                }
}
```