PRÁCTICO 1: MODELOS ESTÁTICOS – INGENIERÍA INVERSA

DIAGRAMAS DE CLASES

Una parte importante dentro del proceso de re-ingeniería de un sistema es la ingeniería inversa del mismo, es decir, la obtención de modelos abstractos a partir del código fuente del sistema. Uno de los modelos a construir es el diagrama de clases correspondiente al diseño de una aplicación que se obtiene a partir del código fuente del sistema al cual se le está realizando la ingeniería inversa.

1.- Construya a partir del código Java dado a continuación un **modelo de diseño** usando diagramas de clases en UML:

```
import java.io.IOException;
import java.lang.*;

public class PruebaNombre {

  public static void main(String args[]) {
    Nombre nombre = new Nombre();
    int opcion = -1;
    while (opcion != 0) {
            mostrarOpciones();
            opcion = -1;
            while (opcion < 0 || opcion > 12)  opcion = selectOpcion();
            switch (opcion) {
                    case 1:   nombre = ingresarNombreCompleto(); break;
                    case 2    nombre = new Nombre(); break;
                    case 3:   System.out.println("\nEl titulo es: " + nombre.getTitulo() + "\n"); break;
                    case 4:   System.out.println("\nEl primer nombre es : " + nombre.getPrimerNombre() + "\n"); break;
                    case 5:   System.out.println("\nEl segundo nombre es: " + nombre.getSegundoNombre() + "\n"); break;
                    case 6:   System.out.println("\nEl apellido es: " + nombre.getApellido() + "\n"); break;
                    case 7:   System.out.println("\nEl nombre completo es: " + nombre.getNombreCompleto() + "\n"); break;
                    case 8:   System.out.println("\nEl nombre abreviado es: " + nombre.getNombreyApellido() + "\n"); break;
                    case 9:   nombre.setTitulo(ingresarEntrada("Nuevo Titulo").trim()); break;
                    case 10:  nombre.setPrimerNombre(ingresarEntrada("Nuevo Primer Nombre").trim());break;
                    case 11:  nombre.setSegundoNombre(ingresarEntrada("Nuevo Segundo Nombre").trim()); break;
                    case 12:  nombre.setApellido(ingresarEntrada("Nuevo Apellido").trim());break;
                    }
            }
} // fin del main

private static void mostrarOpciones() {
System.out.println("Crear Nombre Completo.....1");
System.out.println("Crear Nombre Vacio.....2");
System.out.println("Mostrar Titulo.....3");
System.out.println("Mostrar Primer Nombre.....4");
System.out.println("Mostrar Apellido.....6");
System.out.println("Mostrar Nombre Completo.....7");
System.out.println("Mostrar Nombre y Apellido con inicial...8");
System.out.println("Cambiar Titulo.....9");
System.out.println("Cambiar Primer Nombre.....10");
System.out.println("Cambiar Apellido.....12");
System.out.println("Salir.....0");
}

private static int selectOpcion() {
            . . . . . .
}

private static Nombre ingresarNombreCompleto() {
 return new Nombre(ingresarEntrada("Titulo").trim(), ingresarEntrada("Primer Nombre").trim(), ingresarEntrada("Apellido").trim());
}

private static String ingresarEntrada(String entrada) {
            byte[] s = new byte[20];
            System.out.print("\n" + entrada + ":");
            try {
                    System.in.skip(System.in.available());
                    System.in.read(s);
                    return new String(s);
            }
```

```
		catch (IOException e) {
			System.out.println("Error. No se pudo capturar el " + entrada);
			return new String();
		}
} // class PruebaNombre end

}

import java.lang.*;
public class Nombre{
		private StringBuffer titulo;
		private StringBuffer primerNombre;
		private StringBuffer segundoNombre;
		private StringBuffer apellido;

		public Nombre(String t, String pN, String sN, String a) {
			titulo = new StringBuffer(t);
			primerNombre = new StringBuffer(pN);
			apellido = new StringBuffer(a);
		}
		public Nombre(){
			titulo = new StringBuffer();
			primerNombre = new StringBuffer();
			apellido = new StringBuffer();
		}
		public void setTitulo(String t) {    titulo.replace(0, titulo.length(), t);   }
		public void setPrimerNombre(String pN) {   primerNombre.replace(0, primerNombre.length(), pN);   }
		public void setSegundoNombre(String sN) {    segundoNombre.replace(0, segundoNombre.length(), sN);   }
		public void setApellido(String a) {    apellido.replace(0, apellido.length(), a);    }
		public String getTitulo(){    return titulo.toString();    }
		public String getPrimerNombre(){    return primerNombre.toString();    }
		public String getSegundoNombre(){    return segundoNombre.toString();   }
		public String getApellido(){    return apellido.toString();    }
		public String getNombreCompleto() {
			StringBuffer nombreCompleto = new StringBuffer();
			nombreCompleto.append(getTitulo()+" ");
			if (!getPrimerNombre().equals(""))
				nombreCompleto.append(getPrimerNombre()+" ");
			nombreCompleto.append(getApellido());
			return nombreCompleto.toString().trim();
		}
		public String getNombreyApellido(){
			StringBuffer nomYApe = new StringBuffer();
			nomYApe.append(getPrimerNombre()+" ");
			nomYApe.append(getApellido());
			return nomYApe.toString().trim();
		}
}// class Nombre end
```

2.- Construya a partir del código Java dado a continuación un modelo de diseño estático.

```
import java.lang.*;
public class Gerbil implements Roedor{
		public void saltar() {   System.out.println( "Saltar de Gerbo");   }
		public void comer(){   System.out.println( "Comer de Gerbo");    }
		public void dormir(){   System.out.println( "Dormir de Gerbo");   }
}

import java.lang.*;
public class Hamster implements Roedor{
		public void comer(){   System.out.println( "Comer de Hamster");   }
		public void dormir(){   System.out.println( "Dormir de Hamster");   }
		public void saltar() {   System.out.println( "");   }
}

import java.lang.*;
public interface Roedor{
		public void comer();
		public void dormir();
		public void saltar();
}
```

```
import java.lang.*;
public class Tester{
	public static void main(String args[]){
		Roedor arrayOfRoedores[] = new Roedor[2];
		arrayOfRoedores[0] = new Gerbil();
		arrayOfRoedores[1] = new Hamster();
		for (int i = 0 ; i < arrayOfRoedores.length ; i++) {
			arrayOfRoedores[i].saltar();
			arrayOfRoedores[i].comer();
			arrayOfRoedores[i].dormir();
		}
}  }
```

3.- Construya a partir del código Java dado a continuación un modelo de diseño estático:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
import java.io.*;

public class GUItest extends JFrame implements ActionListener {
	JMenuBar menubar;
	JMenu filemenu;
	JMenuItem quit, reset;
	JButton button1, button2, button3; //Define button variables
	JButton button4;
	JTextField tf1, tf2, tf3, tf4; //Define textfield variable
	JComboBox cb;

	int numberOfTimePressed = 0;
	int numberOfBChanges = 0;
	int numberOfFChanges = 0;

	public GUItest() {
		menubar = new JMenuBar(); //Collects all menus
		filemenu = new JMenu("File"); //A "File"-menu
		reset = new JMenuItem("Reset"); //An "Reset" option
		reset.addActionListener(this);
		filemenu.add(reset); //Add "Reset" to the "File"-menu
		quit = new JMenuItem("Quit"); //An option that can be placed in the file menu (for example)
		quit.addActionListener(this);
		filemenu.add(quit); //Add "Quit" to the "File"-menu
		menubar.add(filemenu); //Add "File"-menu to the menubar
		filemenu.setMnemonic('F');
		quit.setMnemonic('Q');
		reset.setMnemonic('R');
		setJMenuBar(menubar); //set the menubar
		button1 = new JButton("Red"); //create new JButton with text: Red
		button1.addActionListener(this); //Listen if anyone presses button1
		button2 = new JButton("Green");
		button2.addActionListener(this);
		button4 = new JButton("Blue");
		button4.addActionListener(this);
		button3 = new JButton("Quit");
		button3.addActionListener(this);
		tf1 = new JTextField(20); //create new textfield of size 20
		tf1.setText("You have pressed a button " + Integer.toString(numberOfTimePressed) + " times."); //set initial text
		tf1.setFont(new Font("Dialog", Font.BOLD, 14)); //set font
		tf1.setForeground(Color.white); //set color of the foreground
		tf1.setBackground(Color.red); //set color of the background
		tf1.setEditable(false); //It's forbidden to edit the textfield
		tf2 = new JTextField(20); //create new textfield of size 20
		tf2.setText("You have changed the background color " + numberOfBChanges +
			" times and you have changed the foreground color " + numberOfFChanges + " times"); //set initial text
		tf2.setFont(new Font("Dialog", Font.BOLD, 14)); //set font
		tf2.setForeground(Color.white); //set color of the foreground
		tf2.setBackground(Color.red); //set color of the background
		tf2.setEditable(false); //It's forbidden to edit the textfield
		tf3 = new JTextField(20); //create new textfield of size 20
		tf3.setText(""); //set initial text
		tf3.setFont(new Font("Dialog", Font.BOLD, 14)); //set font
		tf3.setForeground(Color.black); //set color of the foreground
		tf3.setBackground(Color.white); //set color of the background
```

```
                tf3.setEditable(true);
                tf3.addActionListener(this);
                tf3.setActionCommand("tf3");
                tf4 = new JTextField(20); //create new textfield of size 20
                tf4.setText("No text has been input"); //set initial text
                tf4.setFont(new Font("Dialog", Font.BOLD, 14)); //set font
                tf4.setForeground(Color.black); //set color of the foreground
                tf4.setBackground(Color.white); //set color of the background
                tf4.setEditable(false);
                String[] elements = { "Change background color", "Change foreground color", "Nothing Hill happen"};
                cb = new JComboBox(elements);
                cb.addActionListener(this);
                cb.setActionCommand("cb");
                //Add the buttons and the textfield according to a layout, GridLayout
                JPanel top = new JPanel(new GridLayout(0, 1));
                . . . . . .
        }
        public void actionPerformed(ActionEvent e) {
                String command = e.getActionCommand();
                if (command.equals("Red"))  { . . . . . }
                else if (command.equals("Green")) { . . . . . }
                else if (command.equals("Blue")) {  . . . . . }
                else if (command.equals("tf3"))    . . . . .
                else if (command.equals("Reset")) { . . . . . .}
                else if (command.equals("Quit")) {  . . . . . . . }
}
//main program

 public static void main(String[] args) {
        GUItest appl = new GUItest(); //create an instance of class GUItest
        appl.setVisible(true);
 }   }
```

4. Dado el siguiente código Java, construya un diagrama de clase que incluya clases, asociaciones, atributos, métodos, visibilidad de métodos, atributos y asociaciones.

```
Class Movie {
        private String _title;
        private Price _price;

        public Movie (String name, Price price) {
                _title = name;
                _price = price;
        }
        int getFrequentRenterPoints(int daysRented) {   return _price.getFrequentRenterPoints(daysRented);   }
        double getCharge(int daysRented) {    return _price.getCharge(daysRented);     }
}
class Rental {
        private Movie _movie;
        private int _daysRented;

        public Rental(Movie movie, int daysRented) {
                _movie = movie;
                _daysRented = daysRented;
        }
        public int getDaysRented() {   return _daysRented;     }
        public Movie getMovie() {     return _movie;     }
        public double getCharge() {     return _movie.getCharge (_daysRented);   }
        public int getFrequentRenterPoints() {   return _movie.getFrequentRenterPoints(_daysRented)   }
}
class Customer {
        private String _name;
        private Vector _rentals = new Vector();

        public Customer(String name) {    _name = name;    }
        public void addRental(Rental arg) {    _rentals.addElement(arg);   }
        public String getName() {    return _name;   }
        statement() {  }
        htmlStatement() {
                Enumeration rentals = _rentals.elements();
                String result = "<H1> Rentals for >EM>" + getName() + "</EM></H1><P>\n";
                While (rentals.hasMoreElements()) {
                        Rental each = (Rental) rentals.nextElement();
```

```
                        Result += each.getMovie().getTitle() + ": " + String.valueOf(each.getCharge()) + "<BR>\n";
                    }
                     . . . . . .
                    return result;
            }
            getTotalCharge() {}
            getTotalFrequentRenterPoints() {
            . . . . . .
            }
}

abstract class Price {
            abstract int GetPriceCode();
            abstract double getCharge(int daysRented);
            int getFrequentRenterPoints(int daysRented) {   return 1;  }
}
Class ChildrensPrice extends Price {
            double getCharge(int daysRented) {  . . . . . }
}
Class NewReleasePrice extends Price {
            double getCharge(int daysRented) {      return daysRented * 3;   }
            int getFrequentRenterPoints(int daysRented) {  . . . .   }
}
Class RegularPrice extends Price {
            double getCharge(int daysRented) {  . . . .   }
}
```

5. Dado el siguiente código Java, construir el diagrama de clase correspondiente.

```
 abstract class EmployeeType {
   abstract int payAmount(Employee emp);
}

class Salesman extends EmployeeType {
   int payAmount(Employee emp) {     return emp.getMonthlySalary() + emp.getCommission();    }
}

class Manager extends EmployeeType {
   int payAmount(Employee emp) {      return emp.getMonthlySalary() + emp.getBonus();    }
}

class Engineer extends EmployeeType {
   int payAmount(Employee emp) {   return emp.getMonthlySalary();    }
}

class Employee {
   private Set _employeeTypes = newHashSet();

   public Set getEmployeeTypes() {   return _employeeTypes;     }
   public void addEmployeeTypes (EmployeeType c) {    __employeeTypes.add(c);      }
   public void removeEmployeeTypes (EmployeeType c) {    _employeeTypes.remove(c);     }
   public int allPayToEmployeeTypes () {
      Set s = Employee.getEmployeeTypes ();
      Iterator iter = s.iterator();
      int total = 0;
      while (iter.hasNext()) {
         EmployeeType each = (EmployeeType) iter.next();
         total += each.payAmount;
      }
     return total;
   }
```

6. Construir el diagrama de clases a partir del siguiente código JAVA.

```
public class App {
   private static ClaseA a;
   public static void main(
      String[] args){
        a = new ClaseA();
        a.am1();
        a.am2();
   }
}


public class ClaseA{
    private ClaseB b;
    private int c;
    public void am1() {
          c = 1;
          b = new ClaseB();
          b.bm1(this);
          show();
    }
    public void am2() {
         …
    }
    public void show() {
      System.out.println(c);
    }
}

public class ClaseB{
    private ClaseA a;
    public void bm1(
     ClaseA unA) {
          a = new ClaseA();
          a.show();
          unA.show();
          a = unA;
          a.show();
          unA.show();
        }
}
```