Linkia FP

Formación Profesional Oficial a Distancia

Actividad

Desarrollo de aplicaciones web
Desarrollo de aplicaciones multiplataforma

M03 – Programación II

Estructuras de datos avanzadas y control de excepciones

Objetivos

- Escribir programas que manipulen información seleccionando y utilizando los tipos avanzados de datos facilitados por el lenguaje.
- Utilizar las clases básicas para almacenar y procesar información (listas y tablas de Hash).
- Implementar la gestión de excepciones en la utilización de clases facilitadas por el lenguaje.
- Implementar el lanzamiento de excepciones.

Actividad 1

Productos (HashMap)

Crear un programa que simule el stock de productos de una tienda. Cada producto consta de un nombre y un precio. Se debe crear una clase Producto que guarde la información especificada. Además, hay que sobrescribir los métodos equals y hashCode de la siguiente forma:

- Dos productos se consideran iguales si tienen el mismo nombre y el mismo precio.
- El hashCode se calcula sumando al precio del producto el código hash del nombre (recuerda que los Strings tienen el método hashCode ya implementado).

El programa principal debe hacer uso de un HashMap para almacenar la siguiente información: los productos serán la clave, y como valor pondremos el stock en la tienda (número entero). Implementar métodos para dar de alta productos, eliminar, modificar y borrar la lista de contactos. No hace falta que realices un menú, simplemente puedes escribir un main en el que se cree el HashMap, se añadan productos, se eliminen y se modifiquen. Haz algún ejemplo de operación con dos productos que el HashMap debe identificar como iguales.

Actividad 2

Garaje mecánico (ArrayList)

Se desea realizar una aplicación que permita a los mecánicos de un garaje registrar, consultar y actualizar los trabajos (reparaciones y revisiones) que han sido realizados o que están en proceso de realización en el garaje.

Cada trabajo se identifica inequívocamente por su "identificador de trabajo". El "identificador de trabajo" es un número que se asocia con el trabajo en el momento que se registra. El primer trabajo registrado tendrá el identificador 0, el segundo el 1 y así sucesivamente.

Los trabajos incluyen una pequeña descripción de la reparación o revisión a realizar.

Todos los trabajos incluyen el número de horas que van siendo necesarias para su realización. Al crear un trabajo el número de horas es 0.El número de horas irá aumentando a medida que los mecánicos van dedicando tiempo a realizar la reparación o la revisión. Cuando el trabajo se ha finalizado se marca como "finalizado" y el número de horas no puede volver a cambiarse.

Las reparaciones incluyen el precio del material utilizado (piezas o pintura). Al registrar una reparación el precio del material es 0 y va aumentando a medida que los mecánicos van utilizando material en la reparación. Una vez que la reparación se marca como "finalizada" no se puede cambiar el precio del material utilizado.

El precio a cobrar para cada trabajo se compone de una parte fija que resulta de multiplicar el número de horas empleadas por 30€. Además, dependiendo del tipo de trabajo el coste varía de la siguiente manera:

- Reparación mecánica: su precio se calcula como fijo más el coste material multiplicado por 1.1.
- Reparación de chapa y pintura: su precio se calcula como fijo más el coste material multiplicado por 1.3.
- Revisión: su precio se calcula como fijo más extra independiente del número de horas de 20€.

La aplicación tendrá las siguientes funcionalidades:

- Registrar trabajo: se introduce el tipo de trabajo y su descripción. El mecánico introduce los datos y el programa añade el trabajo a la lista y muestra el identificador asignado al trabajo.
- Aumenta horas: el mecánico introduce el identificador del trabajo y el número de horas.

- Aumenta coste piezas: el mecánico introduce el identificador del trabajo y el coste de las piezas, y la aplicación aumenta el coste de las piezas al trabajo.
- Finaliza trabajo: el mecánico introduce el identificador del trabajo y la aplicación termina el trabajo.
- Muestra trabajo: el mecánico introduce el identificador del trabajo y la aplicación muestra el identificador, la descripción y el precio del trabajo.

Siempre que el mecánico introduzca un identificador incorrecto, el programa informará de este error.

Utilizar un ArrayList para almacenar los trabajos. Hay que hacer uso de clases y herencias.

Actividad 3

Control de excepciones

1. Crear una clase Alumno con los siguientes campos: nombre y edad. Se debe crear el constructor y los métodos get y set necesarios. Crear una excepción propia DemasiadosObjetos que muestre el mensaje “Se ha alcanzado el número máximo de elementos”. Crear otra clase con el método main que almacene en un ArrayList varios alumnos. Ir haciendo inserciones de forma que si se intenta insertar un sexto alumno, se lance una excepción de tipo DemasiadosObjetos.

2. Ejecuta el siguiente programa:

Archivo Persona.java:

```
public class Persona {
   private String nombre;
   private int edad;

   public Persona(String nombre, int edad) {
      super();
      this.nombre = nombre;
      this.edad = edad;
   }

   public String getNombre() {
      return nombre;
   }

   public void setNombre(String nombre) {
      this.nombre = nombre;
   }

   public int getEdad() {
      return edad;
   }

   public void setEdad(int edad) {
      this.edad = edad;
   }

   @Override
   public String toString() {
      return "Persona [nombre=" + nombre + ", edad=" + edad + "]";
   }

}
```

Archivo Main.java:

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class Main {

   static BufferedReader br = new BufferedReader(new
InputStreamReader(System.in));

   public static void main(String[] args) {
       String nombre;
       int edad;
       try {
          nombre = pedirNombre();
          edad = pedirEdad();
          Persona p = new Persona(nombre, edad);
          System.out.println(p);
       }
       catch(Exception e) {
          System.out.println("ERROR AL CREAR LA PERSONA");
       }
   }
   private static String pedirNombre() {
       System.out.println("Introduce el nombre de la persona:");
       String nombre = "";
       try {
          nombre = br.readLine();
          if(nombre.matches(".*\\d.*")){
              throw new Exception();
          }
       }
       catch(Exception e){
          System.out.println("ERROR AL INTRODUCIR EL NOMBRE");
       }
       return nombre;
   }
   private static int pedirEdad() {
       System.out.println("Introduce la edad de la persona:");
       int edad = 0;
       try {
          edad = Integer.parseInt(br.readLine());
       } catch (Exception e) {
          System.out.println("ERROR AL INTRODUCIR LA EDAD");
       }
       return edad;
   }
}
```

Verás que, aunque se produzca un error al introducir el nombre o la edad de la persona, ésta se crea. Haz los cambios pertinentes en el programa para que, si se produce un error, el programa se detenga y no se cree un objeto de la clase Persona.