Programación, Curso 2015-2016

Práctica 11: Practicar con recorridos y búsquedas en ArrayList y el tratamiento de excepciones

Datos personales
Apellidos: Prieels
Nombre: Cedric

1

2 Código Java de la clase NoEncontrado

```
public class NoEncontrado extends Exception {}
```

3 Código Java de la clase Coche

```
import java.util.ArrayList;
import java.util.Scanner;
import java.util.Locale;
import java.io.*;
import fundamentos.Dibujo;
import fundamentos.ColorFig;
/**
 * Clase para simular y probar el fucnionamiento del sistema de
 * deteccion de colisiones
 */
public class Coche {

   private ArrayList<Obstaculo> lista; //una lista de obstaculos
   private double vC;                  // velocidad del coche en m/s
   private final double A,L;           // dimensiones del coche
                                       // (semianchura y longitud) en m
   /**
```

```
    * Constructor que recibe como parametros la velocidad del coche
    * vC en m/s, las dimensiones A y L del coche en m y el nombre de
    * un fichero de texto del que se leen datos para rellenar la
    * lista de obstaculos
    */
   public Coche(double vC, double A, double L, String nombreFichero) {
      this.vC = vC;
      this.A = A;
      this.L = L;
      lista = new ArrayList<Obstaculo>();
      try (Scanner in=new Scanner(new FileReader(nombreFichero))) {
         in.useLocale(Locale.ENGLISH);
         int id;
         double vT,vN,d,alfa,r;
         in.nextLine();
         while (in.hasNext()) {
            id = in.nextInt();
            vT = in.nextDouble();
            vN = in.nextDouble();
            d = in.nextDouble();
            alfa = in.nextDouble();
            r = in.nextDouble();
            Obstaculo nuevo = new Obstaculo(id,r);
            nuevo.set(vT,vN,d,alfa);
            lista.add(nuevo);
         }
      } catch (FileNotFoundException e) {
         System.out.println("No encuentro el fichero");
         // Error grave. Se abandona el programa
         System.exit(-1);
      }
   }
```

```
/**
 * Retorna un array conteniendo todos los obstaculos para los que se
 * detecta un posible choque
 */
public Obstaculo[] posiblesChoques() {

    ArrayList <Obstaculo> chocan = new ArrayList();

    for (int i = 0; i < lista.size(); i++) {

        boolean choca = false;

        if (lista.get(i).tAlcance(vC) > 0.0
        && lista.get(i).tAlcance(vC) < 30.0
        && lista.get(i).tRebase(vC, L) > 0.0
        && lista.get(i).tRebase(vC, L) < 30.0) {

            if (Math.abs(lista.get(i).margenAlcance(vC)) <= lista.get(i).getRadio() + A) {

                choca = true;

            } else if (Math.abs(lista.get(i).margenRebase(vC, L)) <= lista.get(i).getRadio() + A) {

                choca = true;

            } else if ((lista.get(i).margenAlcance(vC) * lista.get(i).margenRebase(vC, L)) < 0.0) {

                choca = true;

            }

        }
```

```
            if (choca) {

                chocan.add(lista.get(i));

            }

        }

        Obstaculo[] posibles = new Obstaculo[chocan.size()];

        for (int i = 0; i <= posibles.length - 1 ; i++) {

            posibles[i] = chocan.get(i);

        }

        return posibles;

    }

    /**
     * Pone en pantalla un informe de todos los obstaculos
     */
    public void informe() {

        System.out.printf("%15s %15s %15s %15s %15s %n","Id", "tiempoAlcance",
"tiempoRebase", "margenAlcance", "margenRebase");

        for (int i = 0; i < lista.size(); i ++) {

            System.out.printf("%15s %15.2f %15.2f %15.1f %15.1f %n", lista.get(i).getId(),
lista.get(i).tAlcance(vC), lista.get(i).tRebase(vC, L), lista.get(i).margenAlcance(vC),
lista.get(i).margenRebase(vC, L));
```

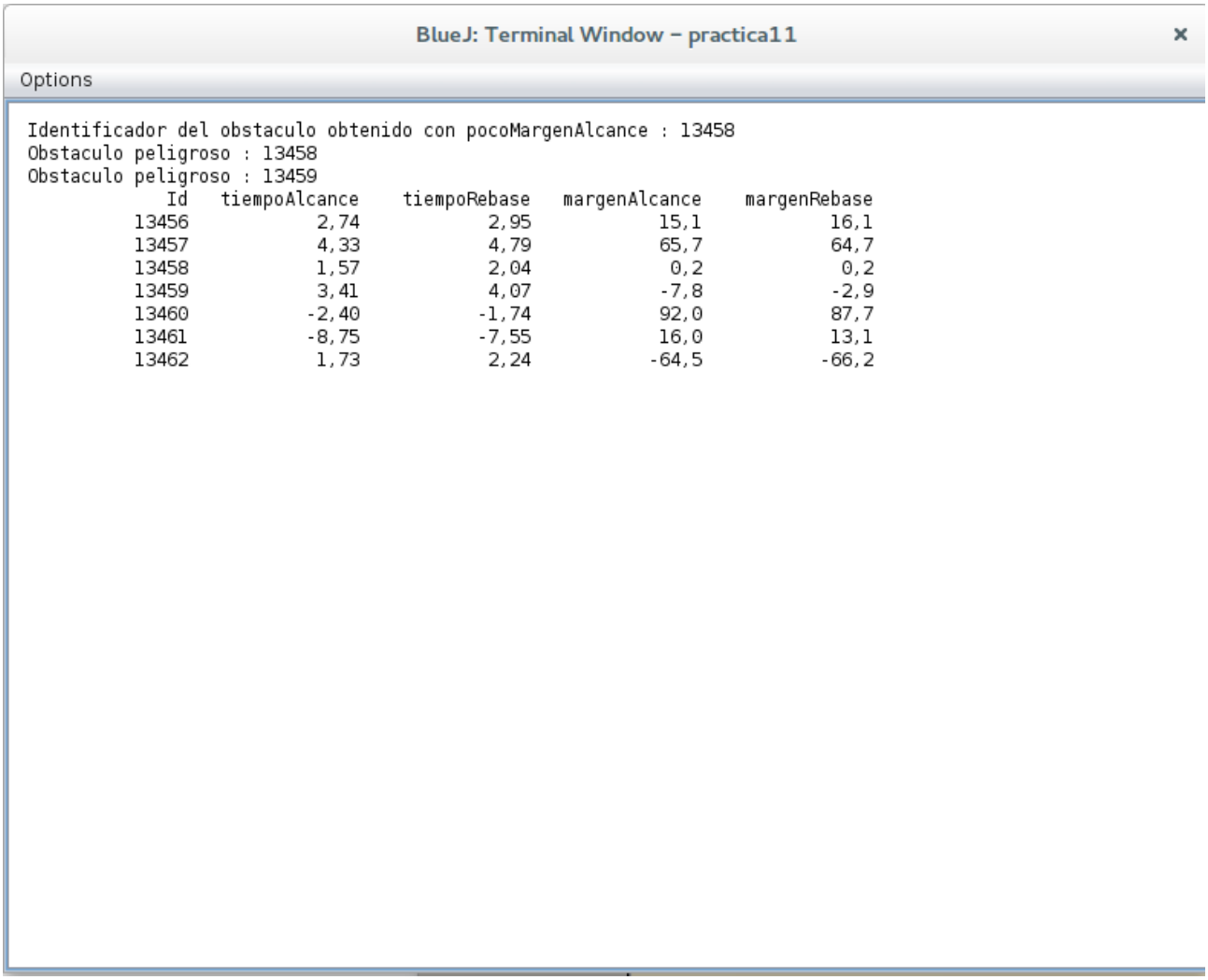
```
        }

    }

    /**
     * Busca en la lista el primer Obstaculo cuyo margen de alcance en
     * valor absoluto es menor o iguar que r+A y lo retorna.  Si no lo
     * encuentra lanza NoEncontrado.
     */
    public Obstaculo pocoMargenAlcance() throws NoEncontrado {

        //try {

            for (int i = 0; i < lista.size(); i++) {

                if (lista.get(i).margenAlcance(vC) < lista.get(i).getRadio() + A) {

                    return lista.get(i);

                }

            }

        // } catch(NoEncontrado e) {

        //    System.out.println("No encontrado");

        // }

        return lista.get(0);

    }
}
```

4 Código Java de la clase que contiene el programa principal

```
/**
 * Clase OperaCoche permite usar los métodos de las clases Obstaculo y Coche para pintar el coche,
 * los obstaculos y tener mas informaciones sobre el problema.
 *
 * @author Cedric Prieels
 * @version 9 de mayo de 2016
 */
public class operaCoche
{

   public static void main() {

      Coche miCoche = new Coche(18.5, 1.3, 4.5, "obstaculos.txt");

         try {

               Obstaculo miObstaculo;
               miObstaculo = miCoche.pocoMargenAlcance();

               System.out.println("Identificador del obstaculo obtenido con pocoMargenAlcance :
"+miObstaculo.getId());

               for (int i = 0; i < miCoche.posiblesChoques().length; i++) {

                  System.out.println("Obstaculo peligroso : "+miCoche.posiblesChoques()[i].getId());

               }
```

```
        } catch (NoEncontrado e) {

            System.out.println("No se ha encontrado ningún obstaculo peligroso");

        } finally {

            miCoche.informe();

        }

    }

}
```

5 Captura de pantalla de la salida obtenida en la consola de Java



6 Código Java del nuevo método (parte avanzada)

7 Captura de pantalla de la ruta (parte avanzada)