PROGRAMACIÓN ORIENTADA A OBJETOS
(Laboratorio de Prácticas)

Titulaciones de Grado en Ingeniería de Computadores, Ingeniería Informática y Sistemas de Información

Sesión 6
EXCEPCIONES EN JAVA
(Soluciones a los ejercicios)

1. Ejercicio Media

Reescribir el ejercicio de la ***"Media"*** de la Sesión 2, de forma que se controle mediante excepciones si el usuario introduce una letra diferente a A, B, C, D y E o F (para finalizar), imprimiendo un mensaje de ***"nota introducida no valida".***

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

public class MediaEx {

    public static void main(String args[]) throws IOException {
        double media;
        int contador, total;
        String nota="";
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));

        total = 0;
        contador = 0;
        do {
            try {

                System.out.print("Teclee calificación (A,B,C,D,E), F para terminar: ");
                nota = entrada.readLine().toUpperCase();

                switch (nota) {
                    case "A":
                        total += 4;
                        contador++;
                        break;
                    case "B":
                        total += 3;
                        contador++;
                        break;
                    case "C":
                        total += 2;
                        contador++;
                        break;
                    case "D":
                        total += 1;
                        contador++;
                        break;
                    case "E":
                        total += 0;
                        contador++;
                        break;
                    case "F":
                        System.out.println("Fin petición calificaciones.");
```

```
                        break;
                    default:
                        throw new MediaException();
                }
            } catch (MediaException | IOException e) {
                System.out.println(e.toString());
            }
        } while (!nota.equals("F"));

        if (contador != 0) {
            media = (double) total / contador;
            System.out.println("\nEl promedio del grupo es: " + media);
        } else {
            System.out.println("\nNo se introdujeron calificaciones.");
        }

    }
}

class MediaException extends Exception {

    public MediaException() {
        super("Excepción: Nota introducida no valida.");
    }
}
```

2. Ejercicio Calculadora

Escribir un programa llamada Calculadora que realice las funciones típicas de una calculadora. Para ello pedirá al usuario tres valores: operación (+,-,*,/), operando1, operando2 y a partir de ellos muestre el resultado de la operación. Cuando el usuario introduzca una Z como valor de la operación el programa parará. Si se introduce un carácter no permitido, se debe producir una excepción definida por el usuario imprimiendo un mensaje de error.
Se deberán intentar utilizar todos los mecanismos de tratamiento de excepciones dados en teoría.

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

public class Calculadora {

    public static void main(String[] args) {
        // TODO code application logic here
        double op1 = 0, op2 = 0, total = 0;
        String op = "";
        BufferedReader entrada = new BufferedReader(new InputStreamReader(System.in));
        while (true) {
            try {
                System.out.println("\nTeclee operación (+,-,*,/), Z para terminar: ");
                op = entrada.readLine().toUpperCase();

                if (!op.equals("+") && !op.equals("-")
                        && !op.equals("*") && !op.equals("/")
                        && !op.equals("Z")) {
                    throw new OpMal();
                }

                if (op.equals("Z")) {
                    return;
                }
                System.out.println("\nTeclee operando 1: ");
                op1 = Double.parseDouble(entrada.readLine());
                System.out.println("\nTeclee operando 2: ");
                op2 = Double.parseDouble(entrada.readLine());
                if (op.equals("+")) {
                    total = op1 + op2;
                }
                if (op.equals("-")) {
```

```
                    total = op1 - op2;
                }
                if (op.equals("*")) {
                    total = op1 * op2;
                }
                if (op.equals("/")) {
                    total = op1 / op2;
                }
                System.out.println("\nEl resultado de " + op1 + op + op2 + " es: " + total);
            } catch (OpMal om) {
                System.out.println("\n" + om.getMessage());
            } catch (NumberFormatException nfe) {
                System.out.println("\nError de formato numérico: " + nfe.toString());
            } catch (ArithmeticException ae) {
                System.out.println("\nError aritmético: " + ae.toString());
            } catch (IOException ioe) {
                System.out.println("\nError de entrada/salida: " + ioe.toString());
            } catch (Exception e) {
                // Captura cualquier otra excepción
                System.out.println(e.toString());
            }
        }
    }
}

class OpMal extends Exception {

    public OpMal() {
        super("Excepción: OPERACIÓN INCORRECTA.");
    }
}
```

3. Ejercicio Publicaciones

Se desea desarrollar un programa para la gestión de publicaciones en una biblioteca. Todas las publicaciones deben tener asociado un autor (String), un título (String) y un año de publicación (int). Deben considerarse tres tipos de publicaciones distintos: libros, revistas y tesis doctorales. En el caso de los libros se debe agregar el nombre de la editorial correspondiente (String). Las revistas deben añadir el número (int) y el volumen (int). Por su parte, las tesis doctorales deberán incluir la Universidad (String) y la calificación (int). Se deben codificar las clases Publicación, Libro, Revista y Tesis con los atributos mencionados, los métodos get y set correspondientes y al menos un constructor. Además debe generarse una excepción definida por el usuario (FechaMal) cuando se intente establecer como año de publicación un año posterior al existente en el sistema o un año negativo. Se debe tener en cuenta la generación de esta excepción tanto en el método set del atributo correspondiente como en el constructor.

```
package publicaciones;

public class FechaMal extends Exception {
    public static String FECHA_INEXISTENTE = "El año de publicación aún no se ha alcanzado.";
    public static String FECHA_NEGATIVA = "El año de publicación no puede ser negativo.";

    public FechaMal() {
        super("Se ha producido una excepción en la fecha de publicación.");
    }

    public FechaMal(String txt) {
        super(txt);
    }
}
```

```
package publicaciones;

import java.time.LocalDate;

public class Publicacion {

    private String autor;
```

```
    private String titulo;

    private int apubli;

    public Publicacion(String autor, String titulo, int apubli) {
        this.autor = autor;
        this.titulo = titulo;
        if (compruebaFecha(apubli)) {
            this.apubli = apubli;
        } else {
            this.apubli = 0;
        }
    }

    public final boolean  compruebaFecha(int a) {
        LocalDate hoy = LocalDate.now();
        int anno = hoy.getYear();
        try {
            if (a < 0) {
                throw new FechaMal(FechaMal.FECHA_NEGATIVA);
            } else if (a > anno) {
                throw new FechaMal(FechaMal.FECHA_INEXISTENTE);
            }
            return true;
        } catch (FechaMal fm) {
            System.out.println("Excepción: " + fm.getMessage());
            return false;
        }
    }

    public int getApubli() {
        return apubli;
    }

    public void setApubli(int apubli) {
        if (compruebaFecha(apubli)) {
            this.apubli = apubli;
        } else {
            this.apubli = 0;
        }
    }

    public String getTitulo() {
        return titulo;
    }

    public void setTitulo(String titulo) {
        this.titulo = titulo;
    }

    public String getAutor() {
        return autor;
    }

    public void setAutor(String autor) {
        this.autor = autor;
    }

    @Override
    public String toString() {
        return "Publicacion{" + "autor=" + autor + ", titulo=" + titulo + ", apubli=" +
apubli + '}';
    }

}
```

```
package publicaciones;

public class Libro extends Publicacion{

    private String editorial;
```

```
    public Libro(String editorial, String autor, String titulo, int apubli) {
        super(autor, titulo, apubli);
        this.editorial = editorial;
    }

    public String getEditorial() {
        return editorial;
    }

    public void setEditorial(String editorial) {
        this.editorial = editorial;
    }

    @Override
    public String toString() {
        return "Libro{" + super.toString() + " editorial=" + editorial + '}';
    }

}
```

```
package publicaciones;

public class Revista extends Publicacion {

    private int numero;

    private int volumen;

    public Revista(int numero, int volumen, String autor, String titulo, int apubli) {
        super(autor, titulo, apubli);
        this.numero = numero;
        this.volumen = volumen;
    }

    public int getVolumen() {
        return volumen;
    }

    public void setVolumen(int volumen) {
        this.volumen = volumen;
    }

    public int getNumero() {
        return numero;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    @Override
    public String toString() {
        return "Revista{" + super.toString() + " numero=" + numero + ", volumen=" + volumen +
'}';
    }

}
```

```
package publicaciones;

public class Tesis extends Publicacion {

    private String universidad;

    private int calificacion;

    public Tesis(String universidad, int calificacion, String autor, String titulo, int
apubli) {
        super(autor, titulo, apubli);
        this.universidad = universidad;
        this.calificacion = calificacion;
    }
```

```
    public int getCalificacion() {
        return calificacion;
    }

    public void setCalificacion(int calificacion) {
        this.calificacion = calificacion;
    }

    public String getUniversidad() {
        return universidad;
    }

    public void setUniversidad(String universidad) {
        this.universidad = universidad;
    }

    @Override
    public String toString() {
        return "Tesis{" + super.toString() + " universidad=" + universidad + ",
calificacion=" + calificacion + '}';
    }

}
```

```
package publicaciones;

public class PruebaPublicaciones {

    public static void main(String[] args) {
        Libro l1 = new Libro("Prentice Hall", "Eckel", "Piensa en Java", 2007);
        Libro l2 = new Libro("Pearson", "Deitel", "Cómo programar en Java", -2012);

        l2.setApubli(2050);
        l2.setApubli(2012);

        System.out.println(l1.toString());
        System.out.println(l2.toString());
    }

}
```

4. Ejercicio Aparcamiento (ejercicio obligatorio para II y SI, voluntario para IC)

Reescribir el código del ejercicio “Aparcamiento” de la sesión 4 de forma que los métodos intoducirVehiculo y sacarVehiculo puedan producir excepciones en las siguientes circunstancias:

- No se puede introducir un vehículo si ya está dentro o el aparcamiento está lleno.
- No se puede sacar un vehículo que no está dentro del aparcamiento.

La aplicación tiene que hacer uso de la clase AparcamientoException para la utilización de excepciones.

```
public class AparcamientoException extends Exception {

    public static String VEHICULO_DENTRO = "El vehículo ya se encuentra en el aparcamiento.";
    public static String VEHICULO_FUERA = "El vehículo no se encuentra en el aparcamiento.";
    public static String APARCAMIENTO_LLENO = "El aparcamiento está lleno.";

    public AparcamientoException() {
        super("Se ha producido una excepción en la aplicación.");
    }

    public AparcamientoException(String txt) {
        super(txt);
    }
}
```

```
package aparcamiento;
```

```
import java.util.ArrayList;

public class Aparcamiento {

    private static ArrayList<Vehiculo> vehiculos = new ArrayList<Vehiculo>();
    private final static int cantidad = 20;
    private static int numero = 0;

    public static ArrayList<Vehiculo> getVehiculos() {
        return vehiculos;
    }

     public static ArrayList<String> getMatriculasVehiculos() {
        ArrayList<String> matriculas = new ArrayList<>();
        for (Vehiculo v : vehiculos) {
            matriculas.add(v.getMatricula());
        }
        return matriculas;
    }

    public static String intoducirVehiculo(Vehiculo v) {
        try {
            if (vehiculos.contains(v)) {//si esta dentro
                throw new AparcamientoException(AparcamientoException.VEHICULO_DENTRO);
            } else if (numero == cantidad) {
                throw new AparcamientoException(AparcamientoException.APARCAMIENTO_LLENO);
            } else {
                numero++;
                vehiculos.add(v);
                return "Vehículo con matrícula " + v.getMatricula() + " aparcado";
            }
        } catch (AparcamientoException ae) {
            return ae.toString();
        }

    }

    public static String sacarVehiculo(Vehiculo v) {
        try {
            if (!Aparcamiento.getVehiculos().contains(v)) {//no esta dentro
                throw new AparcamientoException(AparcamientoException.VEHICULO_FUERA);
            } else {
                double precio = v.calcularImporte();
                numero--;
                vehiculos.remove(v);
                return "Vehiculo con matrícula " + v.getMatricula() + " retirado con precio:
" + precio + " €";
            }
        } catch (AparcamientoException ae) {
            return ae.toString();
        }

    }

    public static String sacarVehiculo(String matricula) {
        try {
            boolean dentro = false;
            Vehiculo ve = null;
            for (Vehiculo v : vehiculos) {
                if (v.getMatricula().equals(matricula)) {
                    ve = v;
                    dentro = true;
                }
            }
            if (!dentro) {//no esta dentro
                throw new AparcamientoException(AparcamientoException.VEHICULO_FUERA);
            } else {
                double precio = ve.calcularImporte();
                numero--;
                vehiculos.remove(ve);
                return "Vehículo con matrícula " + ve.getMatricula() + " retirado con precio:
" + precio + " €";
            }
```

```
        } catch (AparcamientoException ae) {
            return ae.toString();
        }
    }

}
```

```
package aparcamiento;

import java.time.LocalDateTime;
import java.util.Objects;

public abstract class Vehiculo {

    private String matricula;
    private LocalDateTime fechaEntrada;
    private boolean abono;

    public Vehiculo(String matricula, boolean abono) {
        this.matricula = matricula;
        this.abono = abono;
        this.fechaEntrada = LocalDateTime.now();
    }

    public Vehiculo(String matricula, LocalDateTime fechaEntrada, boolean abono) {
        this.matricula = matricula;
        this.fechaEntrada = fechaEntrada;
        this.abono = abono;
    }

    public String getMatricula() {
        return matricula;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    public LocalDateTime getFechaEntrada() {
        return fechaEntrada;
    }

    public void setFechaEntrada(LocalDateTime fechaEntrada) {
        this.fechaEntrada = fechaEntrada;
    }

    public boolean isAbono() {
        return abono;
    }

    public void setAbono(boolean abono) {
        this.abono = abono;
    }

    public abstract double calcularImporte();

    @Override
    public int hashCode() {
        int hash = 7;
        hash = 31 * hash + Objects.hashCode(this.matricula);
        return hash;
    }

    @Override
    public boolean equals(Object obj) {
        if (this == obj) {
            return true;
        }
        if (obj == null) {
            return false;
        }
        if (getClass() != obj.getClass()) {
            return false;
```

```
        }
        final Vehiculo other = (Vehiculo) obj;
        if (!Objects.equals(this.matricula, other.matricula)) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "Vehiculo{" + "matricula=" + matricula + ", fechaEntrada=" + fechaEntrada + ",
abono=" + abono + '}';
    }

}
```

```
package aparcamiento;

import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;

public class Automovil extends Vehiculo {

    private String tipo;

    public Automovil(String matricula, boolean abono, String tipo) {
        super(matricula, abono);
        this.tipo = tipo;
    }

    public Automovil(String matricula, LocalDateTime fechaEntrada, boolean abono, String
tipo) {
        super(matricula, fechaEntrada, abono);
        this.tipo = tipo;
    }

    public String getTipo() {
        return tipo;
    }

    public void setTipo(String tipo) {
        this.tipo = tipo;
    }

    @Override
    public double calcularImporte() {
        LocalDateTime fechaSalida = LocalDateTime.now();
        long minutos = ChronoUnit.MINUTES.between(this.getFechaEntrada(), fechaSalida);
        double tasa = 0, total = 0;

        switch (tipo) {
            case "turismo":
                tasa = 1.5;
                break;
            case "todoterreno":
                tasa = 2.5;
                break;
            case "furgoneta":
                tasa = 3.5;
                break;
        }

        total = minutos * tasa / 60;

        if (this.isAbono()) {
            total -= (total * 0.4);
        }

        return total;
    }

    @Override
    public String toString() {
```

```
        return super.toString() + " # Automovil{" + "tipo=" + tipo + '}';
    }

}
```

```
package aparcamiento;

import java.time.LocalDateTime;
import java.time.temporal.ChronoUnit;

public class Camion extends Vehiculo {

    private int ejes;

    public Camion(String matricula, boolean abono, int ejes) {
        super(matricula, abono);
        this.ejes = ejes;
    }

    public Camion(String matricula, LocalDateTime fechaEntrada, boolean abono, int ejes) {
        super(matricula, fechaEntrada, abono);
        this.ejes = ejes;
    }

    public int getEjes() {
        return ejes;
    }

    public void setEjes(int ejes) {
        this.ejes = ejes;
    }

    @Override
    public double calcularImporte() {
        LocalDateTime fechaSalida = LocalDateTime.now();
        long minutos = ChronoUnit.MINUTES.between(this.getFechaEntrada(), fechaSalida);
        double tasa = 0, total = 0;

        if (ejes <= 3) {
            tasa = 4.5;
        } else {
            tasa = 6.5;
        }
        total = minutos * tasa / 60;

        if (this.isAbono()) {
            total -= (total * 0.4);
        }

        return total;
    }

    @Override
    public String toString() {
        return super.toString() + " # Camion{" + "ejes=" + ejes + '}';
    }

}
```

```
package aparcamiento;

public class PruebaAparcamientoException {

    public static void main(String[] args) {
        Automovil a1 = new Automovil("1111-AAA", true, "turismo");
        Automovil a2 = new Automovil("2222-BBB", false, "todoterreno");
        Automovil a3 = new Automovil("3333-CCC", false, "furgoneta");
        Camion c1 = new Camion("4444-DDD", true, 4);
        Camion c2 = new Camion("5555-EEE", false, 3);
        Camion c3 = new Camion("6666-FFF", false, 5);
        System.out.println(Aparcamiento.intoducirVehiculo(a1));
        System.out.println(Aparcamiento.intoducirVehiculo(a2));
        System.out.println(Aparcamiento.intoducirVehiculo(a3));
```

```
        System.out.println(Aparcamiento.intoducirVehiculo(a3)); //excepción
        System.out.println(Aparcamiento.intoducirVehiculo(c1));
        System.out.println(Aparcamiento.intoducirVehiculo(c2));
        System.out.println(Aparcamiento.intoducirVehiculo(c3));
        System.out.println(Aparcamiento.intoducirVehiculo(c3)); //excepción

        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());

        System.out.println(Aparcamiento.sacarVehiculo(a1));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(a2));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(a3));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(a3)); //excepción
        System.out.println(Aparcamiento.sacarVehiculo(c1.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo("0000-WWW")); //excepción
        System.out.println(Aparcamiento.sacarVehiculo(c2.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());
        System.out.println(Aparcamiento.sacarVehiculo(c3.getMatricula()));
        System.out.println("Vehículos: "+ Aparcamiento.getMatriculasVehiculos().toString());

    }

}
```