

**Universidad Autonoma de Nuevo León.
Facultad de Ciencias Físico Matemáticas.**

Práctica #07 – Manejo de excepciones.

Laboratorio de Programación Orientada a Objetos.

Alumno: Jeremy Uriel Rossell Segura

Matrícula: 2173396

Materia: Laboratorio de Programación Orientada a Objetos.

Docente: Jorge Alberto Islas Pineda

Compilación

El programa compiló adecuadamente.

The screenshot shows the IntelliJ IDEA IDE with the 'Transporte.java' file open. The code defines an abstract class 'Transporte' and a 'Main' class that creates instances of 'TransporteMaritimo', 'TransporteTerrestre', 'TransporteAereo', and 'TransporteFerroviario'. The 'Run' button is highlighted, and the 'Run' tab is active, showing the execution output. The output indicates that the program ran successfully with exit code 0.

```
import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ArrayList<Transporte> transportes = new ArrayList<>();

        // objetos con distintas capacidades
        Transporte mar = new TransporteMaritimo(id: "BAR-01", capacidad: 60);
        Transporte tierra = new TransporteTerrestre(id: "BUS-22", capacidad: 50);
        Transporte aire = new TransporteAereo(id: "AV-777", capacidad: 80);
        Transporte tren = new TransporteFerroviario(id: "TRN-05", capacidad: 60);
    }
}
```

```
public abstract class Transporte {
    private String id;
    private int capacidad;

    public Transporte(String id, int capacidad) {
        setId(id);
        setCapacidad(capacidad);
    }

    public String getId() {
        return id;
    }
}
```

```
public class TransporteMaritimo extends Transporte {
    public TransporteMaritimo(String id, int capacidad) {
        super(id, capacidad);
    }
}

public class TransporteTerrestre extends Transporte {
    public TransporteTerrestre(String id, int capacidad) {
        super(id, capacidad);
    }
}

public class TransporteAereo extends Transporte {
    public TransporteAereo(String id, int capacidad) {
        super(id, capacidad);
    }
}

public class TransporteFerroviario extends Transporte {
    public TransporteFerroviario(String id, int capacidad) {
        super(id, capacidad);
    }
}
```

```
Run: Main
/usr/java/jdk-25-oracle-x64/bin/java -javaagent:/home/jeremy/.local/share/JetBrains/Toolbox/apps/intellij-idea-ultimate/lib/idea_rt.jar=38601 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8
Navegando por mar....
[!] Transportando 60 pasajeros en Marítimo [BAR-01]
Mantenimiento marino: revisión de casco y motores.
--
Rodando en carretera...
[!] ERROR: [!] Pasajeros (60) exceden capacidad (50) en Terrestre
Mantenimiento terrestre: llantas y frenos.
--
Volando en el aire...
[!] Transportando 60 pasajeros en Aéreo [AV-777]
Mantenimiento aéreo: inspección de turbinas y sistemas.
--
Circulando sobre rieles...
[!] Transportando 60 pasajeros en Ferroviario [TRN-05]
Mantenimiento ferroviario: revisión de vías y locomotora.
--
Process finished with exit code 0
```

Depuración

El programa se depuró adecuadamente.

The screenshot shows the IntelliJ IDEA IDE with the 'Transporte.java' file open. The 'Debug' button is highlighted, and the 'Debug' tab is active, showing the execution output. The output indicates that the program ran successfully with exit code 0.

```
public class Main {
    public static void main(String[] args) {
        ArrayList<Transporte> transportes = new ArrayList<>();

        // objetos con distintas capacidades
        Transporte mar = new TransporteMaritimo(id: "BAR-01", capacidad: 60);
        Transporte tierra = new TransporteTerrestre(id: "BUS-22", capacidad: 50);
        Transporte aire = new TransporteAereo(id: "AV-777", capacidad: 80);
        Transporte tren = new TransporteFerroviario(id: "TRN-05", capacidad: 60);
    }
}
```

```
public abstract class Transporte {
    private String id;
    private int capacidad;

    public Transporte(String id, int capacidad) {
        setId(id);
        setCapacidad(capacidad);
    }

    public String getId() {
        return id;
    }
}
```

```
public class TransporteMaritimo extends Transporte {
    public TransporteMaritimo(String id, int capacidad) {
        super(id, capacidad);
    }
}

public class TransporteTerrestre extends Transporte {
    public TransporteTerrestre(String id, int capacidad) {
        super(id, capacidad);
    }
}

public class TransporteAereo extends Transporte {
    public TransporteAereo(String id, int capacidad) {
        super(id, capacidad);
    }
}

public class TransporteFerroviario extends Transporte {
    public TransporteFerroviario(String id, int capacidad) {
        super(id, capacidad);
    }
}
```

```
Debug: Main
/usr/java/jdk-25-oracle-x64/bin/java -agentlib:jdwp=transport=dt_socket,address=127.0.0.1:59009,suspend=y,server=n -javaagent:/home/jeremy/.local/share/JetBrains/Toolbox/apps/intellij-idea-ultimate/lib/idea_rt.jar=38601 -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8
Connected to the target VM, address: '127.0.0.1:59009', transport: 'socket'
Navegando por mar....
[!] Transportando 60 pasajeros en Marítimo [BAR-01]
Mantenimiento marino: revisión de casco y motores.
--
Rodando en carretera...
[!] ERROR: [!] Pasajeros (60) exceden capacidad (50) en Terrestre
Mantenimiento terrestre: llantas y frenos.
--
Volando en el aire...
[!] Transportando 60 pasajeros en Aéreo [AV-777]
Mantenimiento aéreo: inspección de turbinas y sistemas.
--
Circulando sobre rieles...
[!] Transportando 60 pasajeros en Ferroviario [TRN-05]
Mantenimiento ferroviario: revisión de vías y locomotora.
--
Disconnected from the target VM, address: '127.0.0.1:59009', transport: 'socket'
Process finished with exit code 0
```

Código

Main.java y Transporte.java.

```
1 import java.util.ArrayList;
2
3 public class Main {
4     public static void main(String[] args) {
5         ArrayList<Transporte> transportes = new ArrayList<>();
6
7         // objetos con distintas capacidades
8         Transporte mar = new TransporteMaritimo(id: "BAR-01", capacidad: 100);
9         Transporte tierra = new TransporteTerrestre(id: "BUS-22", capacidad: 50);
10        Transporte aire = new TransporteAereo(id: "AV-777", capacidad: 120);
11        Transporte tren = new TransporteFerroviario(id: "TRN-05", capacidad: 200); //
12
13        // polimorfismo
14        transportes.add(mar);
15        transportes.add(tierra);
16        transportes.add(aire);
17        transportes.add(tren);
18
19        int pasajeros = 60;
20
21        for (Transporte t : transportes) {
22            t.mover();
23            try {
24                t.transportar(pasajeros);
25            } catch (CapacidadExcedidaException e) {
26                System.out.println("[!] ERROR: " + e.getMessage());
27            } finally {
28                if (t instanceof Operable op) {
29                    op.realizarMantenimiento();
30                }
31                System.out.println("--");
32            }
33        }
34    }
35}
```

```
1 public abstract class Transporte { 4 inheritors
2     public void setId(String id) {
3         }
4         this.id = id;
5     }
6
7     public int getCapacidad() {
8         return capacidad;
9     }
10
11    public void setCapacidad(int capacidad) {
12        if (capacidad <= 0) {
13            throw new IllegalArgumentException("[!] La capacidad debe ser mayor que 0");
14        }
15        if (capacidad > 500) {
16            // para evitar valores ilógicos o irreales
17            throw new IllegalArgumentException("[!] Capacidad no válida (>500). Revise los datos");
18        }
19        this.capacidad = capacidad;
20    }
21
22    public abstract String tipo(); 4 implementations
23    public abstract void mover(); 4 implementations
24
25    public void transportar(int pasajeros) throws CapacidadExcedidaException {
26        if (pasajeros > capacidad) {
27            throw new CapacidadExcedidaException(
28                "[!] Pasajeros (" + pasajeros + ") exceden capacidad (" + capacidad + ")");
29        } else {
30            System.out.println("[!] Transportando " + pasajeros + " pasajeros en " + tipo());
31        }
32    }
33}
```

CapacidadExcedidaException, Operable, & BasicExceptions.

```
1 public class CapacidadExcedidaException extends Exception {
2     public CapacidadExcedidaException(String mensaje) {
3         super(mensaje);
4     }
5 }
6
```

```
1 public interface Operable {
2     void realizarMantenimiento();
3 }
4
```

```
1 public class BasicExceptions {
2     public int division(int a, int b) {
3         return a / b;
4     }
5
6     public void age(int a) throws Exception {
7         if (a < 0) {
8             throw new Exception("Algo anda mal");
9         }
10    }
11}
12
13
```

El resto de las clases de los diferentes tipos de transportes.

```
1 public class TransporteAereo extends TransporteBase {
2
3     public TransporteAereo(String id, int capacidad) {
4         super(id, capacidad);
5     }
6
7     @Override
8     public String tipo() {
9         return "Aéreo";
10    }
11
12    @Override
13    public void mover() {
14        System.out.println("Volando en el aire");
15    }
16
17    @Override
18    public void realizarMantenimiento() {
19        System.out.println("Mantenimiento aéreo");
20    }
21 }
22
1 public class TransporteFerroviario extends TransporteBase {
2
3     public TransporteFerroviario(String id, int capacidad) {
4         super(id, capacidad);
5     }
6
7     @Override
8     public String tipo() {
9         return "Ferroviario";
10    }
11
12    @Override
13    public void mover() {
14        System.out.println("Circulando sobre rieles");
15    }
16
17    @Override
18    public void realizarMantenimiento() {
19        System.out.println("Mantenimiento ferroviario");
20    }
21 }
22
1 public class TransporteMaritimo extends TransporteBase {
2
3     public TransporteMaritimo(String id, int capacidad) {
4         super(id, capacidad);
5     }
6
7     @Override
8     public String tipo() {
9         return "Marítimo";
10    }
11
12    @Override
13    public void mover() {
14        System.out.println("Navegando por el mar");
15    }
16
17    @Override
18    public void realizarMantenimiento() {
19        System.out.println("Mantenimiento marítimo");
20    }
21 }
22
1 public class TransporteTerrestre extends TransporteBase {
2
3     public TransporteTerrestre(String id, int capacidad) {
4         super(id, capacidad);
5     }
6
7     @Override
8     public String tipo() {
9         return "Terrestre";
10    }
11
12    @Override
13    public void mover() {
14        System.out.println("Rodando sobre ruedas");
15    }
16
17    @Override
18    public void realizarMantenimiento() {
19        System.out.println("Mantenimiento terrestre");
20    }
21 }
22
```