

## Ejercicios

### 1. Polimorfismo y Excepciones

Considera el siguiente bloque de código:

```
class Animal {
    void makeSound() throws Exception {
        System.out.println("Animal makes a sound");
    }
}
class Dog extends Animal {
    void makeSound() throws RuntimeException {
        System.out.println("Dog barks");
    }
}
public class Main {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        try {
            myDog.makeSound();
        } catch (Exception e) {
            System.out.println("Exception caught");
        }
    }
}
```

¿Cuál sería la salida en consola al ejecutar este código?

1. Dog barks
2. Animal makes a sound
3. Exception caught
4. Compilation error

### 2. Hilos (Threads)

Considera el siguiente bloque de código:

```
class MyThread extends Thread {
    public void run() {
        System.out.println("Thread is running");
    }
}
public class Main {
    public static void main(String[] args) {
        Thread t1 = new MyThread();
        Thread t2 = new MyThread();
        t1.start();
        t2.start();
    }
}
```

¿Cuál sería la salida en consola al ejecutar este código?

1. Thread is running (impreso una vez)
2. Thread is running (impreso dos veces)

- 3. Thread is running (impreso dos veces , en orden aleatorio)
- 4. Compilation error

### 3. Listas y Excepciones

Considera el siguiente bloque de código:

```
import java.util.ArrayList;
import java.util.List;

public class Main {
    public static void main(String[] args) {
        List<Integer> numbers = new ArrayList<>();
        numbers.add(1);
        numbers.add(2);
        numbers.add(3);
        try {
            for (int i = 0; i <= numbers.size(); i++) {
                System.out.println(numbers.get(i));
            }
        } catch (IndexOutOfBoundsException e) {
            System.out.println("Exception caught");
        }
    }
}
```

¿Cuál sería la salida en consola al ejecutar este código?

- 1. 1 2 3 Exception caught
- 2. 1 2 3
- 3. Exception caught
- 4. 1 2 3 4

### 4. Herencia, Clases Abstractas e Interfaces

Considera el siguiente bloque de código:

```
interface Movable {
    void move();
}

abstract class Vehicle {
    abstract void fuel();
}

class Car extends Vehicle implements Movable {
    void fuel() {
        System.out.println("Car is refueled");
    }

    public void move() {
        System.out.println("Car is moving");
    }
}
```

```

}

public class Main {
    public static void main(String[] args) {
        Vehicle myCar = new Car();
        myCar.fuel();
        ((Movable) myCar).move();
    }
}

```

¿Cuál sería la salida en consola al ejecutar este código?

1. Car is refueled Car is moving
2. Car is refueled
3. Compilation error
4. Runtime exception

## 5. Polimorfismo y Sobrecarga de Métodos

Considera el siguiente bloque de código:

```

class Parent {
    void display(int num) {
        System.out.println("Parent: " + num);
    }

    void display(String msg) {
        System.out.println("Parent: " + msg);
    }
}

class Child extends Parent {
    void display(int num) {
        System.out.println("Child: " + num);
    }
}

public class Main {
    public static void main(String[] args) {
        Parent obj = new Child();
        obj.display(5);
        obj.display("Hello");
    }
}

```

¿Cuál sería la salida en consola al ejecutar este código?

1. Child: 5 Parent: Hello
2. Parent: 5 Parent: Hello
3. Child: 5 Child: Hello
4. Compilation error

## 6. Hilos y Sincronización

Considera el siguiente bloque de código:

```

class Counter {
    private int count = 0;
    public synchronized void increment() {
        count++;
    }

    public int getCount() {
        return count;
    }
}

class MyThread extends Thread {
    private Counter counter;

    public MyThread(Counter counter) {
        this.counter = counter;
    }

    public void run() {
        for (int i = 0; i < 1000; i++) {
            counter.increment();
        }
    }
}

public class Main {
    public static void main(String[] args) throws InterruptedException {
        Counter counter = new Counter();
        Thread t1 = new MyThread(counter);
        Thread t2 = new MyThread(counter);
        t1.start();
        t2.start();
        t1.join();
        t2.join();
        System.out.println(counter.getCount());
    }
}

```

¿Cuál sería la salida en consola al ejecutar este código?

1. 2000
2. 1000
3. Variable count is not synchronized
4. Compilation error

## 7. Listas y Polimorfismo

Considera el siguiente bloque de código:

```

import java.util.ArrayList;
import java.util.List;

class Animal {
    void makeSound() {
        System.out.println("Animal sound");
    }
}

```

```

class Dog extends Animal { void makeSound() {
    System.out.println("Bark");
}
}
class Cat extends Animal {
    void makeSound() {
        System.out.println("Meow");
    }
}
public class Main {
    public static void main(String[] args) {
        List<Animal> animals = new ArrayList<>();
        animals.add(new Dog());
        animals.add(new Cat());
        animals.add(new Animal());

        for (Animal animal : animals) {
            animal.makeSound();
        }
    }
}

```

¿Cuál sería la salida en consola al ejecutar este código?

1. Animal sound Animal sound Animal sound
2. Bark Meow Animal sound
3. Animal sound Meow Bark
4. Compilation error

## 8. Manejo de Excepciones y Herencia

Considera el siguiente bloque de código:

```

class Base {
    void show() throws IOException {
        System.out.println("Base show");
    }
}

class Derived extends Base {
    void show() throws FileNotFoundException {
        System.out.println("Derived show");
    }
}

public class Main {
    public static void main(String[] args) {
        Base obj = new Derived();
        try {
            obj.show();
        } catch (IOException e) {
            System.out.println("Exception caught");
        }
    }
}

```

}

¿Cuál sería la salida en consola al ejecutar este código?

1. Base show
2. Derived show \*Si tuviera los imports\*
3. Exception caught
4. Compilation error

## 9. Concurrencia y Sincronización

Considera el siguiente bloque de código:

```
class SharedResource {
    private int count = 0;

    public synchronized void increment() {
        count++;
    }

    public synchronized void decrement() {
        count--;
    }

    public int getCount() {
        return count;
    }
}

class IncrementThread extends Thread {
    private SharedResource resource;
    public IncrementThread(SharedResource resource) {
        this.resource = resource;
    }

    public void run() {
        for (int i = 0; i < 1000; i++) {
            resource.increment();
        }
    }
}

class DecrementThread extends Thread {
    private SharedResource resource;
    public DecrementThread(SharedResource resource) {
        this.resource = resource;
    }

    public void run() {
        for (int i = 0; i < 1000; i++) {
            resource.decrement();
        }
    }
}
```

```

public class Main {
    public static void main(String[] args) throws InterruptedException {
        SharedResource resource = new SharedResource();
        Thread t1 = new IncrementThread(resource);
        Thread t2 = new DecrementThread(resource);
        t1.start();
        t2.start();
        t1.join();
        t2.join();
        System.out.println(resource.getCount());
    }
}

```

¿Cuál sería la salida en consola al ejecutar este código?

1. 1000
2. 0
3. -1000
4. Compilation error

## 10. Generics y Excepciones

Considera el siguiente bloque de código:

```

class Box<T> {
    private T item;

    public void setItem(T item) {
        this.item = item;
    }

    public T getItem() throws ClassCastException {
        if (item instanceof String) {
            return (T) item; // Unsafe cast
        }
        throw new ClassCastException("Item is not a String");
    }
}

public class Main {
    public static void main(String[] args) {
        Box<String> stringBox = new Box<>();
        stringBox.setItem("Hello");
        try {
            String item = stringBox.getItem();
            System.out.println(item);
        } catch (ClassCastException e) {
            System.out.println("Exception caught");
        }
    }
}

```

¿Cuál sería la salida en consola al ejecutar este código?

1. Hello
2. Exception caught
3. Compilation error
4. ClassCastException

11. ..

```
public class Main {
    public static void main(String[] args) {
        Padre objetoPadre = new Padre();
        Hija objetoHija = new Hija();
        Padre objetoHija2 = (Padre) new Hija();
        objetoPadre.llamarClase();
        objetoHija.llamarClase();
        objetoHija2.llamarClase();
        Hija objetoHija3 = (Hija) new Padre();
        objetoHija3.llamarClase();
    }
}

public class Hija extends Padre {
    public Hija() {
        // Constructor de la clase Hija
    }

    @Override
    public void llamarClase() {
        System.out.println("Llame a la clase Hija");
    }
}

public class Padre {
    public Padre() {
        // Constructor de la clase Padre
    }

    public void llamarClase() {
        System.out.println("Llame a la clase Padre");
    }
}
```

Resultado:

1. Llame a la clase Padre  
Llame a la clase Hija  
Llame a la clase Hija  
Error: java.lang.ClassCastException
2. Llame a la clase Padre  
Llame a la clase Hija  
Llame a la clase Hija  
Llame a la clase Hija
3. Llame a la clase Padre



Llame a la clase Hija  
Llame a la clase Hija  
Llame a la clase Padre

12. ..

```
import java.text.NumberFormat;
import java.text.ParseException;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;

public class Ejemplos {
    public static void main(String[] args) {
        Animal uno=new Animal();
        Animal dos=new Dog();
        uno.makeSound();
        dos.makeSound();
        Dog tres=(Dog)new Animal();
        tres.makeSound();
    }
}

class Animal {
    void makeSound() {
        System.out.println("Animal sound");
    }
}

class Dog extends Animal {
    void makeSound() {
        System.out.println("Wau Wau");
    }
}
```

1. Animal sound Wau Wau compilation error
2. Compilation Error
3. Animal sound Wau Wau Animal sound
4. Animal sound

13. ...

```
import java.text.NumberFormat;
import java.text.ParseException;
import java.util.Scanner;
import java.util.ArrayList;
import java.util.List;
import java.lang.*;

public class Ejemplos {
    public static void main(String[] args) {
        Cambios uno=new Cambios();
        int x=1;
        String hola="hola";
```

```

        StringBuilder hola2=new StringBuilder("hola2");
        Integer x2=4;
        uno.makeSound(x, hola);
        uno.makeSound(x2, hola2);
        System.out.println("Cambios?: "+x+", "+hola+", "+x2+", "+hola2);
    }
}

```

```

class Cambios{
    void makeSound(int x, String s) {
        s="cambiando string";
        x=5;
    }

    void makeSound(Integer x,StringBuilder s) {
        x=9;
        s=s.delete(0,s.length());
    }
}

```

1. Compilation error
2. Cambios? : 1, hola, 4 ,
3. Cambios?: 1,hola,4,hola2
4. Cambios?: 5,cambiando string,9,

14. ...

```

interface i1{
    public void m1();
}

```

```

interface i2 extends i1 {
    public void m2();
}

```

```

class animal implements i1,i2 {

```

```

//¿Qué métodos debería implementar la clase animal en este espacio?
}

```

1. solo m1
2. m1 y m2
3. ninguno
4. error compilación

15. ....

```

class Animal {
    void makeSound() throws Exception {
        System.out.println("Animal makes a sound");
    }
}

class Dog extends Animal {
    void makeSound() throws RuntimeException {
        System.out.println("Dog barks");
    }
}

```

```

}
}
public class Main {
    public static void main(String[] args) {
        Animal myDog = new Dog();
        try {
            myDog.makeSound();
        } catch (Exception e) {
            System.out.println("Exception caught");
        }
    }
}

```

¿Cuál sería la salida en consola al ejecutar este código?

1. Dog barks
2. Animal makes a sound
3. Exception caught
4. Compilation error

16. ....

```

import java.util.*;
import java.lang.*;
import java.io.*;

```

```

class Main {

    public static void main(String[] args) {
        String str = "1a2b3c4d5e6f";
        String []splitStr = str.split("//D");

        for(String elemento : splitStr){
            System.out.println(elemento);
        }
    }
}

```