JAVA PROBLEMAS PROPUESTOS (PARTE I)

EJERCICIO 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

La respuesta correcta es 1, a pesar de que el método que tiene lanza un Exception en el main se hace responsable de la excepción con el try/catch, ahora al invocar el método makeSound() por el principio de polimorfismo se invoca el método declarado en la clase Dog sin ninguna excepción por lo que el catch no tiene nada que atrapar.

****

EJERCICIO 2

Ejercicio de 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

****

EJERCICIO 3

Ejercicio de 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

****

EJERCICIO 4

Ejercicio de 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

¿Cuál crees que es la respuesta correcta?

****

EJERCICIO 5

Ejercicio de 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

¿Cuál crees que es la respuesta correcta?

****

EJERCICIO 6

Ejercicio de 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

¿Cuál crees que es la respuesta correcta?

****

EJERCICIO 7

Ejercicio de 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

¿Cuál crees que es la respuesta correcta?

****

EJERCICIO 8

Ejercicio de 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

3- Exception caught

4- Compilation error

¿Cuál crees que es la respuesta correcta?

****

EJERCICIO 9

Ejercicio de 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

¿Cuál crees que es la respuesta correcta?

****

EJERCICIO 10

Ejercicio de 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

¿Cuál crees que es la respuesta correcta?

RESPUESTAS

EJERCICIO 1:

Respuesta correcta opcion 4.

4- Compilation error

La razón es que el método makeSound en la clase Dog tiene una firma diferente respecto a las excepciones lanzadas en comparación con el método makeSound en la clase Animal. En Java, un método sobrescrito no puede lanzar más tipos de excepciones comprobadas que el método en la clase base. En este caso, Dog.makeSound lanza una RuntimeException (que es una excepción no comprobada) mientras que Animal.makeSound lanza una Exception (que es una excepción comprobada). Este conflicto provoca un error de compilación.

EJERCICIO 2:

Respuesta correcta opcion 3

3- Thread is running (impreso dos veces, en orden aleatorio)

La razón es que se están creando y comenzando dos hilos (t1 y t2), cada uno ejecutando el método run de la clase MyThread, lo que imprime "Thread is running". Dado que los hilos pueden ejecutarse en cualquier orden, la salida puede variar, pero la línea "Thread is running" se imprimirá dos veces.

EJERCICIO 3:

Respuesta correcta opcion 1

1- 1 2 3 Exception caught

La razón es que el bucle for está intentando acceder a un índice fuera del rango de la lista. Cuando i llega a numbers.size() (que es 3), numbers.get(i) lanza una IndexOutOfBoundsException, que es capturada en el bloque catch, imprimiendo "Exception caught"

EJERCICIO 4:

Respuesta correcta opcion 1

1- Car is refueled Car is moving

La clase Car extiende la clase abstracta Vehicle e implementa la interfaz Movable. En el método main, el objeto myCar es de tipo Vehicle, pero como es una instancia de Car, el método fuel se ejecuta correctamente. Luego, se hace un casting a Movable para llamar al método move, que también se ejecuta sin problemas.

EJERCICIO 5:

Respuesta correcta opcion 1

1- Child: 5 Parent: Hello

En este caso, el objeto obj es de tipo Parent, pero se instancia como Child. Debido al polimorfismo, el método display(int num) de Child se ejecuta cuando se pasa un entero, y el método display(String msg) de Parent se ejecuta cuando se pasa una cadena, ya que Child no sobrescribe este método.

EJERCICIO 6:

Respuesta correcta opcion 1

1- 2000

La clase Counter tiene un método increment que está sincronizado, lo que significa que solo un hilo puede acceder a él a la vez. Los dos hilos (t1 y t2) ejecutan el método increment 1000 veces cada uno, sumando un total de 2000 incrementos en la variable count. Debido a la sincronización, no hay condiciones de carrera y el valor final de count será 2000.

EJERCICIO 7:

Respuesta correcta opcion 1

2- Bark Meow Animal sound

La lista animals contiene instancias de Dog, Cat y Animal. Cuando el bucle for itera sobre la lista y llama al método makeSound, se ejecuta el método correspondiente a cada instancia. Debido al polimorfismo, el método makeSound de Dog imprime "Bark", el de Cat imprime "Meow" y el de Animal imprime "Animal sound".

EJERCICIO 8:

Respuesta correcta opcion 4

4- Compilation error

El problema es que el método show en la clase Derived lanza una FileNotFoundException, que es una excepción comprobada y más específica que la IOException lanzada por el método show en la clase Base. En Java, un método sobrescrito no puede lanzar excepciones más generales que las que lanza el método en la clase base. Por lo tanto, el código no compila debido a esta incompatibilidad.

EJERCICIO 9:

Respuesta correcta opcion 2

2- 0

En este ejercicio, tanto IncrementThread como DecrementThread acceden a la misma instancia de SharedResource y ejecutan sus respectivos métodos increment y decrement. Ambos métodos están sincronizados, por lo que no hay condiciones de carrera y los incrementos y decrementos están bien sincronizados.

Como el IncrementThread incrementa el contador 1000 veces y el DecrementThread lo decrementa 1000 veces, el resultado final de getCount() será 0, ya que las operaciones se cancelan entre sí.

EJERCICIO 10:

Respuesta correcta opcion 1

1- Hello

Justificacion:

En este código, el método getItem() de la clase Box realiza un casting inseguro de item a String. Sin embargo, dado que en el caso de stringBox se está usando un Box<String> y se ha establecido un String como ítem, el casting es seguro y no se lanza ninguna excepción.

La ClassCastException se lanzaría solo si el tipo de item no fuera String, pero en este caso, el ítem es efectivamente un String, por lo que el método getItem() devuelve "Hello" sin lanzar ninguna excepción.}

Hernández, Alejandro

```
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:

a) Llame a la clase Padre
Llame a la clase Hija
Llame a la clase Hija
Error: java.lang.ClassCastException

b) Llame a la clase Padre
Llame a la clase Hija
Llame a la clase Hija
Llame a la clase Hija

c) Llame a la clase Padre
Llame a la clase Hija
Llame a la clase Hija
Llame a la clase Padre

d) No se UnU

[09:40 a. m.] Hernández, Alejandro

Es la respesta 1

[09:41 a. m.] Hernández, Alejandro

pesimo servicio

```
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

```
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,

```
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
