

No terminado.

Ejercicio 1: Polimorfismo y Excepciones

Considera el siguiente bloque de código:

```
2 usages 1 inheritor
3 class Animal {
4     void makeSound() throws Exception {
5         System.out.println("Animal makes a sound");
6     }
7 }
1 usage
8 class Dog extends Animal {
9     void makeSound() throws RuntimeException {
10        System.out.println("Dog barks");
11    }
12 }
13 public class Main {
14     public static void main(String[] args) {
15         Animal myDog = new Dog();
16         try {
17             myDog.makeSound();
18         } catch (Exception e) {
19             System.out.println("Exception caught");
20         }
21     }
22 }
23 |
24
```

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

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

Respuesta:

Dog barks

Ejercicio 2: Ejercicio de Hilos (Threads)

Considera el siguiente bloque de código:

```
2 usages
3 class MyThread extends Thread {
4     public void run() {
5         System.out.println("Thread is running");
6     }
7 }
8
9 public class Main {
10     public static void main(String[] args) {
11         Thread t1 = new MyThread();
12         Thread t2 = new MyThread();
13         t1.start();
14         t2.start();
15     }
16 }
17 |
```

¿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

Respuesta:

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

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

Ejercicio 3: Ejercicio de Listas y Excepciones

Considera el siguiente bloque de código:

```
3      import java.util.ArrayList;
4      import java.util.List;
5
6      public class Main {
7          public static void main(String[] args) {
8              List<Integer> numbers = new ArrayList<>();
9              numbers.add(1);
10             numbers.add(2);
11             numbers.add(3);
12
13             try {
14                 for (int i = 0; i <= numbers.size(); i++) {
15                     System.out.println(numbers.get(i));
16                 }
17             } catch (IndexOutOfBoundsException e) {
18                 System.out.println("Exception caught");
19             }
20         }
21     }
22
```

¿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

Respuesta:

1 2 3 Exception caught

Ejercicio 4: Ejercicio de Herencia, Clases Abstractas e Interfaces

Considera el siguiente bloque de código:

```

2 usages 1 implementation
4 interface Movable {
    1 usage 1 implementation
5     void move();
6 }
7
8 2 usages 1 inheritor
8 abstract class Vehicle {
    1 usage 1 implementation
9     abstract void fuel();
10 }
11
12 1 usage
12 class Car extends Vehicle implements Movable {
    1 usage
13     void fuel() {
14         System.out.println("Car is refueled");
15     }
16
17     1 usage
17     public void move() {
18         System.out.println("Car is moving");
19     }
20 }
21
22 public class Main {
23     public static void main(String[] args) {
24         Vehicle myCar = new Car();
25         myCar.fuel();
26         ((Movable) myCar).move();
27     }
28 }
29

```

¿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

Respuesta:

Car is refueled Car is moving

Ejercicio 5: Ejercicio de Polimorfismo y Sobrecarga de Métodos

Considera el siguiente bloque de código:

```
2 usages 1 inheritor
3 class Parent {
4     1 usage 1 override
5     void display(int num) {
6         System.out.println("Parent: " + num);
7     }
8
9     1 usage
10    void display(String msg) {
11        System.out.println("Parent: " + msg);
12    }
13 }
14
15 1 usage
16 class Child extends Parent {
17     1 usage
18    void display(int num) {
19        System.out.println("Child: " + num);
20    }
21 }
22
23 public class Main {
24     public static void main(String[] args) {
25         Parent obj = new Child();
26         obj.display(5);
27         obj.display("Hello");
28     }
29 }
```

¿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

Respuesta:

Child: 5 Parent: Hello

Ejercicio 6: Ejercicio de Hilos y Sincronización

Considera el siguiente bloque de código:

```

2  class Counter {
    2 usages
3      private int count = 0;
4
5      1 usage 1 override
6      public synchronized void increment() {
7          count++;
8      }
9
10     1 usage
11     public int getCount() {
12         return count;
13     }
14 }
15
16  2 usages
17  class MyThread extends Thread {
18      2 usages
19      private Counter counter;
20
21      2 usages
22      public MyThread(Counter counter) {
23          this.counter = counter;
24      }
25
26      public void run() {
27          for (int i = 0; i < 1000; i++) {
28              counter.increment();
29          }
30      }
31  }
32
33  public class Main {
34      public static void main(String[] args) throws InterruptedException {
35          Counter counter = new Counter();
36          Thread t1 = new MyThread(counter);
37          Thread t2 = new MyThread(counter);
38          t1.start();
39          t2.start();
40          t1.join();
41          t2.join();
42          System.out.println(counter.getCount());
43      }
44  }

```

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

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

Respuesta:

2000

Ejercicio 7: Ejercicio de Listas y Polimorfismo

Considera el siguiente bloque de código:


```

3  import java.util.ArrayList;
4  import java.util.List;
5
6  class Animal {
7      void makeSound() {
8          System.out.println("Animal sound");
9      }
10 }
11
12 class Dog extends Animal {
13     void makeSound() {
14         System.out.println("Bark");
15     }
16 }
17
18 class Cat extends Animal {
19     void makeSound() {
20         System.out.println("Meow");
21     }
22 }
23
24 public class Main {
25     public static void main(String[] args) {
26         List<Animal> animals = new ArrayList<>();
27         animals.add(new Dog());
28         animals.add(new Cat());
29         animals.add(new Animal());
30
31         for (Animal animal : animals) {
32             animal.makeSound();
33         }
34     }
35 }

```

¿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

Respuesta:

Bark Meow Animal sound

Ejercicio 8: Ejercicio de Manejo de Excepciones y Herencia

Considera el siguiente bloque de código:

```

4 import java.io.FileNotFoundException;
5 import java.io.IOException;
6
7 class Base {
8     void show() throws IOException {
9         System.out.println("Base show");
10    }
11 }
12
13 class Derived extends Base {
14     void show() throws FileNotFoundException {
15         System.out.println("Derived show");
16     }
17 }
18
19 public class Main {
20     public static void main(String[] args) {
21         Base obj = new Derived();
22         try {
23             obj.show();
24         } catch (IOException e) {
25             System.out.println("Exception caught");
26         }
27     }
28 }

```

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

- 1- Base show
- 2- Derived show
- 3- Exception caught
- 4- Compilation error

Respuesta:

Derived show

Ejercicio 9: Ejercicio de Concurrency y Sincronización

Considera el siguiente bloque de código:

```
6 usages
4  class SharedResource {
      3 usages
5      private int count = 0;
6
      1 usage
7      public synchronized void increment() {
8          count++;
9      }
10
      1 usage
11     public synchronized void decrement() {
12         count--;
13     }
14
      1 usage
15     public int getCount() {
16         return count;
17     }
18 }
19
```

```
19      1 usage
20  ✓ class IncrementThread extends Thread {
      2 usages
21      private SharedResource resource;
22
      1 usage
23  ✓ public IncrementThread(SharedResource resource) {
24      |     this.resource = resource;
25      | }
26
27  ⚙️ ✓ public void run() {
28  ✓     |     for (int i = 0; i < 1000; i++) {
29      |         |     resource.increment();
30      |         | }
31      |     }
32  }
33
```

```
1 usage
34  ✓ class DecrementThread extends Thread {
      2 usages
35      private SharedResource resource;
36
      1 usage
37  ✓ public DecrementThread(SharedResource resource) {
38      |     this.resource = resource;
39      | }
40
41  ⚙️ ✓ public void run() {
42  ✓     |     for (int i = 0; i < 1000; i++) {
43      |         |     resource.decrement();
44      |         | }
45      |     }
46  }
```

```
48 ▶ public class Main {  
49 ▶     public static void main(String[] args) throws InterruptedException {  
50         SharedResource resource = new SharedResource();  
51         Thread t1 = new IncrementThread(resource);  
52         Thread t2 = new DecrementThread(resource);  
53         t1.start();  
54         t2.start();  
55         t1.join();  
56         t2.join();  
57         System.out.println(resource.getCount());  
58     }  
59 }
```

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

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

Respuesta:

0

Ejercicio 10: Ejercicio de Generics y Excepciones

Considera el siguiente bloque de código:

```

2 usages
4 class Box<T> {
    3 usages
5     private T item;
6
    1 usage
7     public void setItem(T item) {
8         this.item = item;
9     }
10
    1 usage
11     public T getItem() throws ClassCastException {
12         if (item instanceof String) {
13             return (T) item; // Unsafe cast
14         }
15         throw new ClassCastException("Item is not a String");
16     }
17 }
18
19 ► public class Main {
20 ►     public static void main(String[] args) {
21         Box<String> stringBox = new Box<>();
22         stringBox.setItem("Hello");
23
24         try {
25             String item = stringBox.getItem();
26             System.out.println(item);
27         } catch (ClassCastException e) {
28             System.out.println("Exception caught");
29         }
30     }
31 }

```

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

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

Respuesta:

Hello

Ejercicio 10: Animal Sound

```
4  ▶ public class Main {  
5  ▶     public static void main(String[] args) {  
6      Animal uno=new Animal();  
7      Animal dos=new Dog();  
8      uno.makeSound();  
9      dos.makeSound();  
10     Dog tres=(Dog)new Animal();  
11     tres.makeSound();  
12     }  
13 }  
14  
5 usages 1 inheritor  
15 ⚙ class Animal {  
16     3 usages 1 override  
17     void makeSound() {  
18         System.out.println("Animal sound");  
19     }  
20 }  
3 usages  
21 class Dog extends Animal {  
22     3 usages  
23     void makeSound() {  
24         System.out.println("Wau Wau");  
25     }  
}
```

- 1) Animal sound Wau Wau compilation error
- 2) Comilation Error
- 3) Animal sound Wau Wau Animal sound
- 4) Animal sound

Respuesta:

Animal sound Wau Wau compilation error

Ejercicio 11: Cambios




```
15 import java.lang.*;
16
17 public class Main {
18     public static void main(String[] args) {
19         Cambios uno=new Cambios();
20         int x=1;
21         String hola="hola";
22         StringBuilder hola2=new StringBuilder("hola2");
23         Integer x2=4;
24         uno.makeSound(x, hola);
25         uno.makeSound(x2, hola2);
26         System.out.println("Cambios?: "+x+", "+hola+", "+x2+", "+hola2);
27     }
28 }
29
29 2 usages
30 class Cambios{
31     1 usage
32     void makeSound(int x, String s) {
33         s="cambiando string";
34         x=5;
35     }
36     1 usage
37     void makeSound(Integer x,StringBuilder s) {
38         x=9;
39         s=s.delete(0,s.length());
40     }
41 }
```

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

Respuesta:

Cambios?: 1,hola,4,

Ejercicio 12: Interfaces

```
2 usages 3 implementations
4  interface i1{
    no usages
5     public void m1();
6 }
7
1 usage 2 implementations
8  interface i2 extends i1 {
    no usages
9     public void m2();
10 }
11
no usages
12 public class Animal implements i1,i2 {
13  //¿Qué métodos debería implementar la clase animal en este espacio?
14 }
```

- 1) solo m1
- 2) m1 y m2
- 3) ninguno
- 4) error compilación

Respuesta:

m1 y m2

Ejercicio 13: Clases, y herencia.

```
4 ▶ public class Main {
5
6 ▶     public static void main(String[] args) {
7
8         Padre objetoPadre = new Padre();
9         Hija objetoHija = new Hija();
10        Padre objetoHija2 = (Padre) new Hija();
11        objetoPadre.llamarClase();
12        objetoHija.llamarClase();
13        objetoHija2.llamarClase();
14        Hija objetoHija3 = (Hija) new Padre();
15        objetoHija3.llamarClase();
16
17    }
18 }
```

```
5 usages
3 public class Hija extends Padre {
4     2 usages
5     public Hija() {
6         // Constructor de la clase Hija
7     }
8
9     4 usages
10    @Override
11    public void llamarClase() {
12        System.out.println("Llame a la clase Hija");
13    }
14 }
```

```
6 usages 1 inheritor
3 ▶ public class Padre {
4     3 usages
5     public Padre() {
6         // Constructor de la clase Padre
7     }
8
9     4 usages 1 override
10    public void llamarClase() {
11        System.out.println("Llame a la clase Padre");
12    }
13 }
```

- 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