UNIVERSIDAD DE COSTA RICA

SR-CIE

Carrera de
Informática Empresarial
Sedes Regionales

Curso: IF0004 - Desarrollo de Software II

Docente: Angélica Ulate Céspedes

Práctica de examen I

Importante

El objetivo de esta práctica es brindar a los estudiantes una oportunidad de repasar y reforzar los conceptos vistos en clase.

Es importante destacar que:

1. Esta práctica no garantiza que el examen será idéntico; los ejercicios del examen pueden variar en formato, dificultad o enfoque. Tener presente que para efectos de práctica las instrucciones van a ser muy guiadas.
2. Los estudiantes pueden encontrar temas adicionales o combinaciones de conceptos en el examen que no estén incluidos en esta práctica.
3. Se recomienda utilizar esta práctica como guía para identificar fortalezas y áreas de mejora, pero no sustituye el estudio completo del contenido del curso.
4. Además, considere la opción de practicar los ejercicios que hay en las presentaciones vistas.

OOP

1. Crear la interfaz PuedeCantar (que es capaz de cantar), un interfaz muy simple que sólo posee un método cantar. Crear la clase Persona que implemente el interfaz PuedeCantar y que cuando cante lo haga con las notas musicales. Crear las clases Canario y Gallo que implementen el interfaz PuedeCantar y que muestre cómo cantan. Realizar el programa que haga cantar a un canario, un gallo y un tenor.

Arreglos, manejo de hileras y matrices

1. Crea un programa que pida diez números enteros por teclado, los almacene en un array, y luego muestre todos sus valores.

2. Leer 5 números por teclado y a continuación realizar la media de los números positivos, contar los negativos y contar el número de ceros, y luego mostrar la información que se obtuvo, la media de los positivos, cuántos negativos y cuántos 0 .

3. Dado dos arreglos de 10 números enteros cada uno, crea un tercer arreglo de 20 posiciones que combina los elementos de los dos arreglos de forma intercalada. El arreglo resultante debe almacenar primero un elemento del arreglo 1, luego uno del arreglo 2, después nuevamente uno del arreglo 1, seguido de uno del arreglo 2, y así sucesivamente hasta completar el arreglo.

4. Leer los datos correspondientes a dos arreglos de 12 elementos numéricos cada una. A partir de ellos, crear una tercer arreglo de 24 elementos, en la cual se mezclan los valores de los arreglos de la siguiente forma: se deben almacenar tres elementos consecutivos del arreglo 1, seguidos de tres elementos consecutivos del arreglo 2, repitiendo este proceso hasta completar el arreglo 3.

5. Dado un arreglo de 10 elementos de tipo String, desarrolle una aplicación que determine si las palabras almacenadas en el arreglo se encuentran

ordenadas alfabéticamente de forma ascendente, ordenadas alfabéticamente de forma descendente o si se encuentran desordenadas. Use compareTo y recuerde que este método devuelve positivo si la palabra es mayor (b > a) y negativo si la palabra es menor(a < b).

6. Hacer un programa que acepte un String por teclado. Por la salida queremos que nos diga cuántas letras tiene sin contar espacios en blanco. Use charAt

7. Hacer un programa que acepte un String por teclado. Por la salida queremos que nos diga cuántas letras ‘a’ tiene.

8. Se requiere desarrollar un programa en Java que lea una frase ingresada por teclado y determine si dicha frase es un palíndromo. Para realizar esta validación, el programa debe ignorar los espacios en blanco y no debe diferenciar entre letras mayúsculas y minúsculas. Se asume que la frase estará compuesta únicamente por letras y espacios, sin incluir signos de puntuación, números ni caracteres especiales.

   Un palíndromo se define como un texto que se lee de la misma forma tanto de izquierda a derecha como de derecha a izquierda. El programa deberá analizar la frase ingresada y mostrar por pantalla un mensaje indicando si la frase corresponde o no a un palíndromo.

   Se recomienda usar toLowerCase().replace() y charAt()

   Ejemplos de palíndromos:

   - Anita lava la tina
   - Dabale arroz a la zorra el abad
   - Amigo no gima
   - Amo la pacífica paloma
   - A man a plan a canal Panama
   - No traces en ese cartón

9. Verificar si la suma total de todas las filas de una matriz cuadrada es igual a la suma total de todas las columnas.

   Ejemplo:

   {1, 2, 3, 4},

   {5, 6, 7, 8},

   {1, 1, 1, 1},

   {2, 2, 2, 2}

   Suma de filas = 48

   Suma de columnas = 48

10. Se requiere desarrollar un programa que permita crear e inicializar una matriz cuadrada de tamaño NxN con valores previamente definidos en el código. Una vez inicializada la matriz, el programa debe calcular la suma de los elementos de la diagonal principal y la suma de los elementos de la diagonal inversa. Posteriormente, el sistema deberá calcular la diferencia entre ambas sumas e indicar por pantalla el valor de dicha diferencia, mostrando además las sumas correspondientes a cada diagonal.

11.

Arreglos con objetos

1. Hacer un programa que declare un arreglo de objetos Carro e inicialice cuatro carros, cada uno con una marca de coche.El programa debe recorrer el arreglo utilizando un bucle for-each e imprimir por pantalla las marcas de los coches almacenados.

2. Se requiere desarrollar una aplicación en Java para la gestión básica de productos de una tienda. Para ello, se debe crear una clase Producto, cuya única responsabilidad sea representar un producto, incluyendo los atributos nombre y precio. Asimismo, se debe implementar una clase de servicio que contenga un método estático encargado de realizar la búsqueda de un producto dentro de un arreglo de productos, a partir de su nombre. Este

método debe encargarse únicamente de la lógica de búsqueda, sin realizar operaciones de entrada o salida. Finalmente, desde la clase principal se debe crear un arreglo con al menos tres productos, solicitar el nombre del producto a buscar e invocar el método correspondiente para mostrar el resultado por pantalla.

3. Se debe crear una clase llamada Libro, cuya única responsabilidad sea representar un libro. Esta clase debe contener los siguientes atributos: ISBN, título, autor y número de páginas. Además, debe incluir los métodos get y set correspondientes para cada atributo y sobrescribir el método toString() para mostrar la información del libro con el siguiente formato:

   “El libro con ISBN ___, creado por el autor ___, tiene ___ páginas.”

   Adicionalmente, se debe crear una segunda clase encargada de la gestión de los libros, la cual debe administrar un arreglo de objetos Libro. Esta clase debe implementar un método que permita agregar libros al arreglo, recibiendo como parámetros los valores necesarios para crear cada objeto Libro. Dicha clase será responsable únicamente de la lógica relacionada con el manejo de los libros, como el almacenamiento y la comparación entre ellos.

Desde la clase principal (main), se deben crear e insertar al menos dos libros utilizando valores previamente definidos (valores quemados), sin solicitar datos al usuario por teclado. Posteriormente, el programa debe mostrar por pantalla la información de todos los libros almacenados y determinar cuál de ellos posee el mayor número de páginas, indicando el resultado al usuario. Considere validaciones en caso de que el arreglo este lleno, si no hay libros que mostrar.

Otros

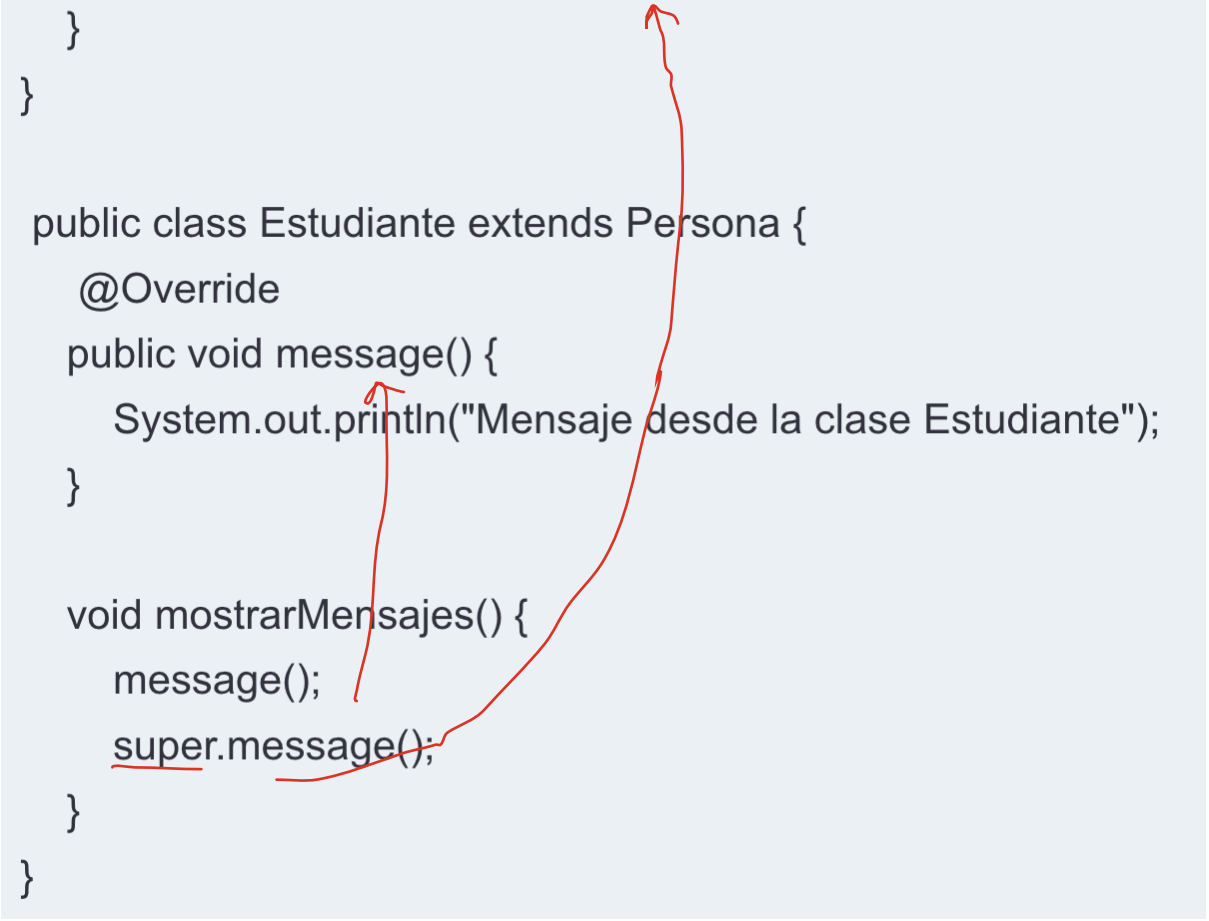
Para facilitar la visibilidad de los ejercicios parece estar en un mismo archivo pero obviamente debemos asumir que están en archivos separados (buenas prácticas).

1. Observe el siguiente fragmento de código

```
public class Persona {
    public void message() {
        System.out.println("Mensaje desde la clase Persona");
    }
}

 public class Estudiante extends Persona {
     @Override
    public void message() {
        System.out.println("Mensaje desde la clase Estudiante");
    }

    void mostrarMensajes() {
        message();
        super.message();
    }
}
```



Cuando se ejecuta el método mostrarMensajes() en la clase Main desde un objeto de la clase Estudiante indique ¿qué imprime o si identifica un error indique cuál es?

"Mensaje desde la clase Estudiante"
"Mensaje desde la clase Persona"

2. Observe el siguiente fragmento de código

```
public class Person {
    Person() {
        System.out.println("Constructor de Person");
    }
}

public class Student extends Person {
    Student() {
        System.out.println("Constructor de Student");
    }
}

public class Main {
 public static void main(String[] args) {
   Student s = new Student();
 }
}
```

¿Qué se imprime en pantalla al ejecutar el siguiente código? o si identifica un error indique ¿cuál es?

Constructor de Person
Constructor de Student

Siempre se ejecuta primero el constructor del padre, luego el del hijo (porque el constructor del hijo llama implícitamente super() como primera línea si tú no escribes otro super(...)).

3. Observe el siguiente fragmento de código

```
public class BirdSeed {
   private int numberBags;
   boolean call;

   public BirdSeed() {
      // LINE 1
      call = false;
      // LINE 2
   }

   public BirdSeed(int numberBags) {
      this.numberBags = numberBags;
   }

   public static void main(String[] args) {
      var seed = new BirdSeed();
      System.out.print(seed.numberBags);
   } }
```

Cuál código puede ser insertado para obtener un print 2

Opciones

A. Reemplazar la línea 1 por BirdSeed(2);

B. Reemplazar la línea 2 por BirdSeed(2);

C. Reemplazar la línea 1 por new BirdSeed(2);

D. Reemplazar la línea 2 por new BirdSeed(2);

E. Reemplazar la línea 1 por this(2);

F. Reemplazar la línea 2 por this(2);

G. El código imprime 2 sin realizar ningún cambio.

5. ¿Qué pares de modificadores pueden usarse juntos en la declaración de un método?

A. static y final

B. private y static

C. static y abstract

D. private y abstract

E. abstract y final

F. private y final

6. ¿Cuáles de las siguientes afirmaciones sobre los métodos son verdaderas? (Seleccione todas las que correspondan.)

A. Los métodos sobrecargados (overloaded) deben tener la misma firma.

B. Los métodos sobrescritos (overridden) deben tener la misma firma.

C. Los métodos sobrecargados (overloaded) deben tener el mismo tipo de retorno.

D. Los métodos sobrescritos (overridden) deben tener el mismo tipo de retorno.

7. ¿Cuál es la salida del siguiente programa?

```
1:  class Mammal {
2:     private void sneeze() {}
3:     public Mammal(int age) {
4:        System.out.print("Mammal");
5:     } }
6:  public class Platypus extends Mammal {
7:     int sneeze() { return 1; }
8:     public Platypus() {
9:        System.out.print("Platypus");
10:    }
11:    public static void main(String[] args) {
12:       new Mammal(5);
13:    } }
```

Opciones:

- Platypus
- Mammal
- PlatypusMammal
- MammalPlatypus
- El código compilará si se cambia la línea 7.
- El código compilará si se cambia la línea 9: *agregando super(<int>); como primera línea del constructor de Platypus*

8. ¿Cuál de las siguientes opciones completa el constructor para que este código imprima 50?

```
class Speedster {
   int numSpots;
}
public class Cheetah extends Speedster {
   int numSpots;

   public Cheetah(int numSpots) {
      // INSERT CODE HERE
   }

   public static void main(String[] args) {
      Speedster s = new Cheetah(50);
      System.out.print(s.numSpots);
   }
}
```

Opciones:

1. numSpots = numSpots;
2. numSpots = this.numSpots;
3. this.numSpots = numSpots;
4. numSpots = super.numSpots;
5. super.numSpots = numSpots;
6. El código no compila independientemente del código insertado en el constructor.
7. Ninguna de las anteriores.

9. ¿Cuál es la salida del siguiente código?

```
1:  class Arthropod {
2:     protected void printName(long input) {
3:        System.out.print("Arthropod");
4:     }
5:     void printName(int input) {
6:        System.out.print("Spooky");
7:     } }
8:  public class Spider extends Arthropod {
9:     protected void printName(int input) {
10:       System.out.print("Spider");
11:    }
12:    public static void main(String[] args) {
13:       Arthropod a = new Spider();
14:       a.printName((short)4);
15:       a.printName(4);
16:       a.printName(5L);
17:    } }
```

Opciones:

A. SpiderSpiderArthropod

B. SpiderSpiderSpider

C.SpiderSpookyArthropod

D.SpookySpiderArthropod

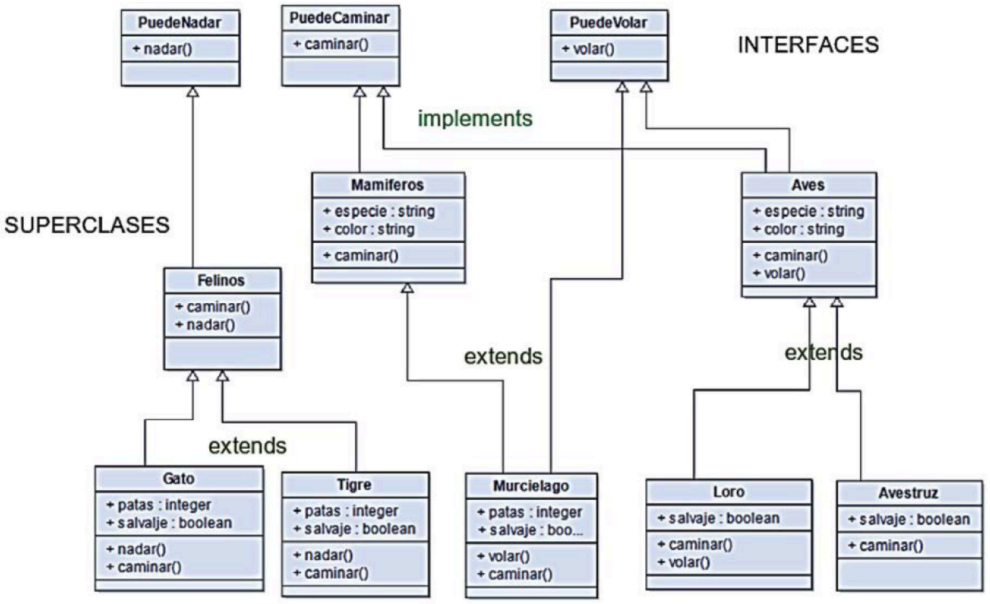
E.El código no compila por la línea 5.

F.El código no compila por la línea 9.

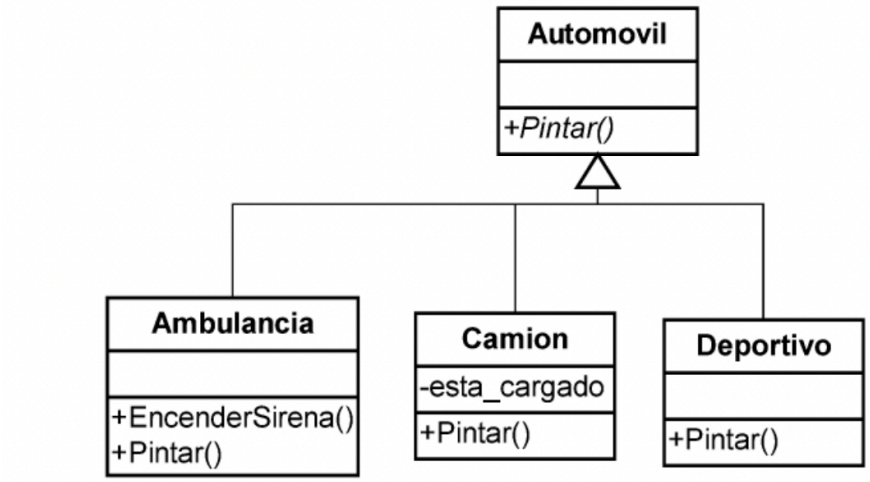
G. Ninguna de las anteriores.

UML

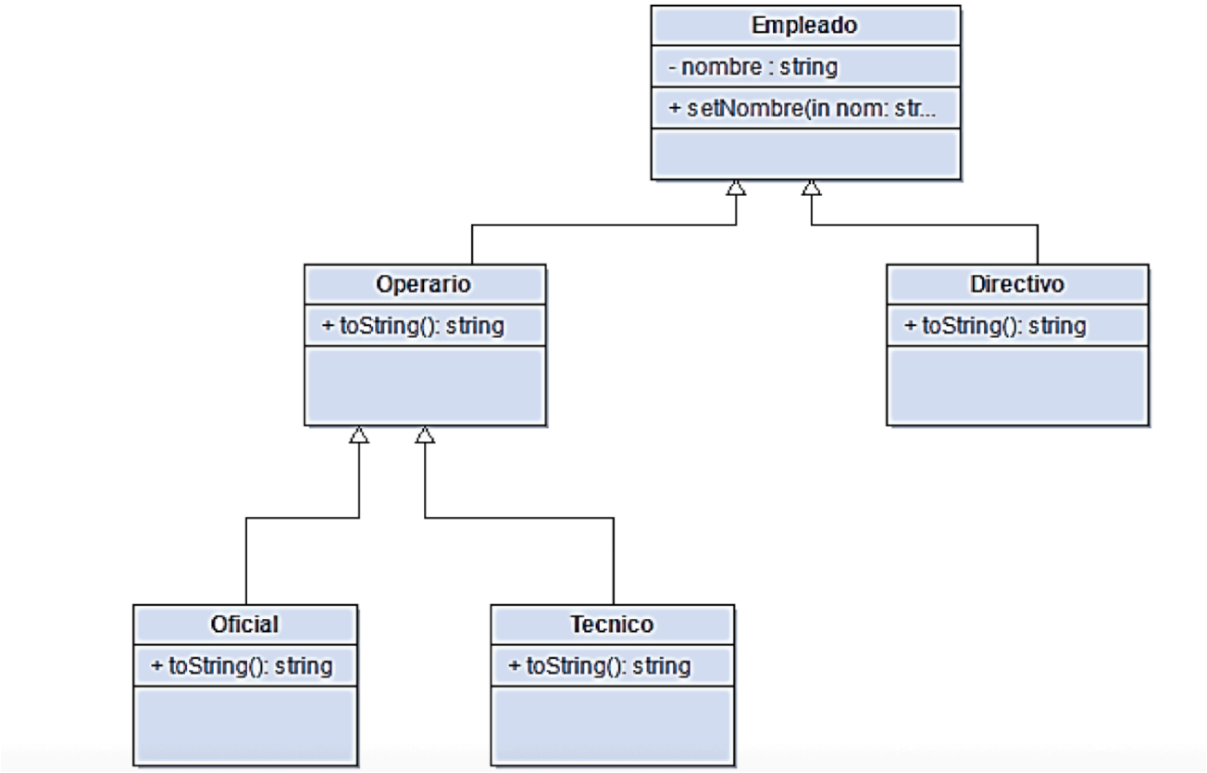
1. Interprete el siguiente diagrama de clases y pásalo a código según se indica. En este caso el ejemplo no trae las flechas discontinuas pero indica que es una implementación.



2. Interprete el siguiente diagrama de clases y pásalo a código según se indica. En este caso el ejemplo no trae las flechas discontinuas pero indica que es una implementación.



3. Interprete el siguiente diagrama de clases y pásalo a código según se indica. En este caso el ejemplo no trae las flechas discontinuas pero indica que es una implementación.



Bubblesort

Escriba el método burbuja para que ordene, practique de forma ascendente y descendente. Puede realizarlo con números, la idea es que sepa como se comporta. Por ejemplo, en la iteración cuando la i vale 3 y j vale 2, como va ordenado el arreglo hasta el momento.

Pendientes
búsqueda
Principios
….