Departamento de Computación
FCEFQyN, Universidad Nacional de Río Cuarto
Asignatura: Estructuras de Datos y Algoritmos - Algoritmos y Estructuras de Datos II
Primer Cuatrimestre de 2025

Práctica No. 4

Para los siguientes ejercicios proveer test para todos los métodos que agregue para chequear que satisfacen las especificaciones.
Acceder al código de base de los ejercicios de esta práctica aceptando el siguiente assignment: `https://classroom.github.com/a/8IbKXnF-`.

1. Ejercicios para resolver en el proyecto *network-v3*

   1.1 Crea objetos de las clases *MessagePost*, *PhotoPost* y *NewsFeed* e invoca sus métodos.

   1.2 Agregue una clase *EventPost* que permita manejar publicaciones de eventos estándar. Los eventos comunes podrían incluir que un usuario se una a un grupo, se haga amigo de otro usuario o cambie su foto de perfil.

   1.3 ¿Tuvo que modificar alguna de las clases existentes para incorporar EventPosts?

2. Escriba un programa Java para crear una clase llamada *Shape* con los métodos *getPerimeter()* y *getArea()*. Cree una subclase llamada *Circle* que sobrescriba los métodos *getPerimeter()* y *getArea()* para calcular el área y el perímetro de un círculo. Agregue además las subclases *Rectangle* y *Square*.

3. Cree un proyecto para modelar el funcionamiento en una universidad, que cuenta con las clases *Person*, *Teacher*, *Student* y *PhDStudent*. Las clases *Teacher* y *Student* son subclases de *Person* y *PhDStudent* es una subclase de *Student*.
   Responda primero intuitivamente las siguientes preguntas y luego realice tests en el proyecto para verificar las respuestas:

   3.1 ¿Cuáles de las siguientes asignaciones son legales?. Aclare por qué si/no valen.
   - Person $p1$ = new Student();
   - Person $p2$ = new PhDStudent();
   - PhDStudent $phd1$ = new Student();
   - Teacher $t1$ = new Person();
   - Student $s1$ = new PhDStudent();

   3.2 Supongamos que tenemos las siguientes declaraciones y aserciones legales:
   - Person $p1$ = new Person();
   - Person $p2$ = new Person();
   - PhDStudent $phd1$ = new PhDStudent();
   - Teacher $t1$ = new Teacher();
   - Student $s1$ = new Student();

   Teniendo en cuenta lo anterior, ¿cuáles de las siguientes asignaciones son legales?. Aclare por qué si/no valen.
   - $s1 = p1$;
   - $s1 = p2$;
   - $p1 = s1$;
   - $t1 = s1$;
   - $s1 = phd1$;
   - $phd1 = s1$;

3.3 Agregue una clase *Curso* al proyecto universidad que contenga una lista de estudiantes. Agregue a esta clase, un método:

```
public boolean find(Student s1)
```

que permita consultar si un estudiante determinado está dentro del curso.

4. Dado el siguiente fragmento de código:

```
Device dev = new Printer();
dev.getName();
```

Donde *Printer* es subclase de *Device*.

- ¿Cuál de estas clases debe tener una definición del método *getName* para que el código compile?
- En la misma situación que en el inciso anterior, si ambas clases tienen una implementación de *getName*, ¿cuál se ejecutará?

5. Supongamos que escribe una clase *Student* sin una superclase declarada y no escribe un método *toString*.

- Considere las siguientes líneas de código:

```
Student st = new Student();
String s = st.toString();
```

¿Considera que el código compila? ¿Qué pasa si se ejecuta?

- Siguiendo con el ejercicio, ¿compilan las siguientes líneas de código?:

```
Student st = new Student();
System.out.println(st);
```

- Supongamos que la clase *Student* sobreescribe el método *toString* para que devuelva el nombre del estudiante. Ahora tiene una lista de estudiantes. ¿compilará el siguiente código? Si no, ¿por qué? Si es así, ¿qué imprimirá? Explique detalladamente qué sucede.

```
for(Object st : myList) {
    System.out.println(st);
}
```

6. Escriba unas pocas líneas de código que den como resultado una situación donde una variable x tiene el tipo estático T y el tipo dinámico D.

7. Lanzar excepciones cuando los métodos de ClockDisplay violen su precondición. Escribir tests negativos para estos métodos (tests que verifican que cuando el método se invoca en un estado inválido lanza una excepción).

8. Utilizando el template del proyecto *music-organizer-v5* del repositorio, lance excepciones en los métodos de la clase *MusicOrganizer* que violen la precondición.

9. Completar ejercicios de manejo de excepciones, vistos en la teoría, en el proyecto *weblog-analyzer* y escriba test negativos para los diferentes casos de excepciones esperadas.

.