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. 1

Parte de esta práctica cuenta con esquemas de programación que pueden accederse a partir del classroom de github, para el cual es conveniente tener una cuenta en Github (`http://github.com`).

1. Utilizando los comandos de git apropiados acceda al template del código fuente, correspondiente a la práctica 1, a través del siguiente enlace: https://classroom.github.com/a/0hdxvhqM .

2. Acceda al proyecto *lab-classes* que incluye la implementación de las clases *LabClass* y *Student* y realizar las siguientes actividades en la clase *Main*:

   2.1 Cree varios objetos de la clase Student, pasando parámetros al constructor que permitan representar algunos estudiantes reales de la materia.

   2.2 Inspeccione el perfil de los diferentes métodos de la clase Student y describa el tipo de retorno de los métodos *getName* y *addCredits*.

   2.3 Crear una instancia de la clase *LabClass* por cada comisión de esta materia, proporcione la cantidad máxima de estudiantes que permite el curso creado (considere para asignar este valor, la capacidad máxima de estudiantes de las salas de computación).

   2.4 Analice los diferentes campos(fields) de la clase *LabClass* e indique los tipos de cada campo.

   2.5 Utilice el método *enrollStudent* para agregar los estudiantes creados a las comisiones que les corresponden en la materia. Analice los campos y métodos de esta clase, para asignar un profesor a cada comisión, un aula y horario, con los datos reales de las comisiones correspondientes.

   2.6 ¿Cual es el tipo de los objetos que se pasan como parámetro al método *enrollStudent*?

   2.7 invoque al método *printList* de las instancias de LabClass, para obtener los detalles de cada comisión.

3. Acceda al código de base del proyecto *my-ticket-machine*. Para cada una de las siguientes actividades edite los comentarios de los métodos, agregando, cuando sea necesario, la precondición usando *@pre* y la poscondición usando *@post*.

   3.1 Agregue un método llamado `showPrice` a la clase *TicketMachine*. El método no toma parámetros y debe retornar void. El cuerpo del método debe imprimir la siguiente línea:

   The price of a ticket is xyz cents.

   Donde `xyz` debe reemplazarse por el valor guardado en el campo `precio` cuando se llama al método.

   3.2 Crear dos objetos de la clase *TicketMachine* con diferentes precios de ticket. ¿Las llamadas al método *showPrice* muestran el mismo resultado o son diferentes? ¿Cómo explica este efecto?.

   3.3 Agregue un constructor de la clase *TicketMachine* para que no tome parámetros. En su lugar, el precio de las entradas debe fijarse en 1000 centavos. Pruebe la implementación creando objetos de la clase *TicketMachine* a través de los diferentes constructores. ¿Qué efecto tiene esto cuando construye objetos de *TicketMachine*?

   3.4 Implemente un método, *empty*, que simule el efecto de retirar todo el dinero de la máquina. Este método debe retornar void y su cuerpo simplemente debe establecer el campo total en cero. ¿Este método necesita tomar algún parámetro? Pruebe el método creando una máquina, insertando algo de dinero, imprimiendo algunos boletos, verificando el total y luego vaciando la máquina. ¿El método *empty* es modificador u observador?

4. Siguiendo la estructura de paquetes de los proyectos de los ejercicios anteriores, cree un nuevo proyecto *heater-exercise*. Dentro del proyecto, cree una clase *Heater*, que contiene sólo un atributo *temperature* de tipo punto flotante de precisión doble (investigue los tipos en Java que se adecúe a la descripción provista). Defina además un constructor que no tome parámetros que se encargue de inicializar el atributo *temperature* con el valor 15.0. Defina los métodos modificadores *warmer* y *cooler*, cuyo efecto es incrementar o decrementar, respectivamente, el valor de *temperature* en 5.0° .

5. Modifique la clase *Heater* del ejercicio previo y defina tres nuevos atributos de tipo punto flotante de precisión doble: *min*, *max* e *increment*. Los valores de *min* y *max* deberían ser inicializados utilizando valores pasados como parámetros en el constructor, mientras que *increment* debe ser inicializado con el valor 5.0. Modificar los métodos *warmer* y *cooler* de manera que utilicen el vlor de *increment* en lugar del valor fijo 5.0.
   Luego de verificar que la primera parte del ejercicio funcione correctamente, modifique el método *warmer* de manera que no permita actualizar el valor de *temperature* a un valor mayor que *max*. Análogamente modificar el método *cooler* de manera que no permita actualizar *temperature* con un valor menor que *min*.
   Agregue un método *setIncrement* que toma un sólo parámetro y lo usa para inicializar el valor de *increment*. Cree una clase *Main* con el método *main*, construya objetos de la clase *Heater* y utilice sus métodos.