Práctica 2. Constructores

Objetivo: Aplicar los conceptos básicos de la programación orientada a objetos en un lenguaje de programación.

Introducción:

Constructor

Hasta ahora, hemos estado creando instancias e inicializando objetos mediante el uso de la palabra clave "new" y lo que parece ser un método con el mismo nombre de la clase que declaró para el objeto (Ej. MyClass ();). Puede o no tener argumentos dentro de sus paréntesis, pero lo que es importante recordar en este momento es que este "método" se denomina constructor y su propósito es inicializar los atributos de ese objeto después de que haya sido instanciado el objeto.

```
MyObject obj = new MyObject();
```

Los constructores solo se pueden invocar una vez cuando se usa la palabra clave "new" para crear una instancia de un objeto. No podemos llamar al constructor cada vez que necesitamos cambiar los valores dentro de un objeto en particular. Para hacer eso, debemos usar los métodos de establecimiento de clases. Todas las clases tienen un constructor predeterminado, que inicializa cada atributo del objeto a "0", "null" o "false" dependiendo de su tipo cuando se llama. Si queremos que nuestro objeto tenga valores iniciales diferentes para sus atributos, necesitamos crear nuestros propios constructores dentro de la definición de clase. Para hacerlo, debe declarar algo similar a un nuevo método dentro de su clase, utilizando un modificador de acceso (comúnmente público), sin ningún modificador de valor de retorno y exactamente el mismo nombre que la clase en la que se encuentra. Luego puede asignar los atributos de clase cualquier valor que desee dentro del cuerpo de nuestro nuevo constructor.

Aquí hay un ejemplo de cómo un constructor inicializa los atributos de un objeto:

```
public MyObject() {
    name = "Maria";
    id = 1202;
}
```

Los constructores generalmente se declaran con el modificador de acceso público, pero pueden usar cualquiera de ellos (predeterminado, protegido, etc.). Es importante saber qué paquetes necesitarán para acceder a la clase para que puedan instanciar objetos de dicha clase.

/* *Ejercicio 1*: Implementa una clase con un constructor privado y ve que sucede. */

Los constructores tampoco tienen un tipo de retorno. Su único trabajo es crear una instancia de la clase, asignando suficiente memoria para el objeto resultante e inicializando sus atributos. La creación del objeto se maneja en tiempo de ejecución, lo que significa que un objeto no se instanciará si el programa no puede acceder a esa parte del código (Ej. Dentro de un "if" que nunca se convierte en verdadero).

Sin embargo, es posible que no siempre queramos tener los mismos atributos en cada objeto que creamos esa clase. En cuyo caso, en su lugar, definimos un constructor que recibe esos valores iniciales diferentes como argumentos y los asigna a los atributos de la clase.

/**Ejercicio 2*: Has un constructor con parámetros. */

El siguiente constructor recibe dos argumentos y los usa para inicializar los atributos.

```
public MyObject(String _name, int _id) {
    name = _name;
    id = _id;
}
```

Tal vez se esté preguntando acerca de esos guiones bajos al comienzo de los nombres de los parámetros. Si usáramos el mismo nombre para los parámetros y los atributos de las clases, cada operación o asignación dentro del constructor se haría con las variables de los parámetros en lugar de los atributos de las clases. Esto se debe a que los parámetros del método "sombrean" los atributos de las clases, lo que hace que esos atributos sean inalcanzables.
/* *Ejercicio 3*: Has un constructor con parámetros que usen los mismos nombres que los atributos de la clase, un método que imprima los valores de las clases y vea si los valores se asignaron correctamente. */

"this" keyword
El uso de diferentes nombres para los parámetros del constructor y los atributos de clase parece una solución fácil para este problema, pero puede hacer que el código sea más difícil de seguir y provocar errores. Afortunadamente, Java nos proporciona una palabra clave conocida como "this".

```
public MyObject(String name, int id) {
    this.name = name;
    this.id = id;
}
```

Al agregar el prefijo "this." (tenga en cuenta el punto) a un nombre de variable, le estamos diciendo a Java que nos estamos refiriendo al atributo de la clase, en lugar del parámetro declarado dentro del paréntesis. El resultado de las expresiones en el ejemplo anterior será que los valores pasados al constructor se asignarán a los atributos.
/**Ejercicio 4*: Modifique el constructor que realizó en el último ejercicio agregando la palabra clave * "this" y verifique si los resultados son diferentes. */

Practica:

1. Crear una clase llamada "Estudiante" que contenga un nombre, identificación del estudiante, materia, calificación y estado.
2. Si la calificación dada es inferior a 60, marque el estado del alumno como reprobado, caso contrario aprobado.
3. Cree 5 objetos de estudiante ingresando sus datos en los argumentos del constructor.
4. Utilizando el constructor y métodos creados en el punto anterior, solicitar al usuario los datos de 5 alumnos. El usuario deberá introducir los datos para cada alumno a capturar (nombre, identificación, materia, calificación y estado).
5. Imprimir la información de los 10 alumnos (punto 3 y 4).

Actividades:

- Hacer un reporte con portada de hoja completa que incluya el código fuente y una captura de pantalla de la práctica funcionando.
- Si la práctica contiene menú, se agregará una captura de pantalla por cada opción implementada.
- Adjuntar en el reporte evidencia de la realización de los ejercicios planteados en esta práctica.
- Subir a blackboard código fuente y reporte.