Introducción a Ciencias de la Computación
Tarea 4: Scanner y Controles de flujo

Bonilla Ruiz Roberto Adrián

Fecha de entrega: 13 de abril 2020

✜ Preguntas: (Primera parte)

1. ¿Para qué sirven los parámetros del main *"String arg []"* ?

   Que debe aceptar un argumento de método de tipo arreglo *"String"*,pues tomando en cuenta los apuntes de clase, *"String"* representa el tipo de parámetro a recibir, mientras que *"arg"* ó *"args"* es el nombre del parámetro, el cual puede tener cualquier nombre siempre y cuando este sea válido. Así mismo, investigando en diversas fuentes se aclara que:

   El método main() acepta un parámetro (y solo uno), es decir: Una matriz de tipo String. La cual toma los valores que son introducidos a la hora de ejecutarse desde la línea de comandos, aún así da igual el valor introducido porque el JRE lo convierte a un String.

   Fuentes: `https://javautodidacta.es/metodo-main-en-java/`
   `https://www.asesoriaensig.com.mx/blog/el-metodo-main-en-java/`

2. Escribe 5 métodos de Scanner que nos ayuden a tomar los datos de entrada. Agrega su definición del API de JAVA

Sintaxis del método	Definición del API de JAVA (resumido)
`int nextInt()`	Recibe un número entero leído del teclado.
`byte nexByte()`	Recibe un byte leído del teclado.
`float nextFloat()`	Recibe un número real grande leído del teclado.
`short nextShort()`	Recibe un número entero pequeño leído del teclado.
`boolean nextBoolean()`	Recibe un booleano leído del teclado.

3. ¿En qué momento se dispara *java.util.InputMismatchException*?

   Cuando el tipo de dato de entrada que se espera no es el recibido.

4. Observa y analiza los siguientes códigos:

NOTA

Si hay error de sintaxis o semántica, ¿Como se corrige?
Deja tu resultado y el resultado de la ejecución, explica cual fue tu error.

a)

```
int i = 0;
while(i++ < 4){
    System.out.println(i);
}
System.out.println("valor final "+i);
```

b)

```
int i = 3;
while(i++ < 4){
    System.out.println(i);
}
System.out.println("valor final "+i);
```

c)

```
for(int i = 0 ; i < 4 ; i ++){
    System.out.println(i);
}
System.out.println("valor final "+i);
```

d)

```
for(int i = 0, j = 0; i < 4 && j<3; i+=j){
    System.out.println(i);
}
System.out.println("valor final "+j);
```

4.1 ¿Cuántas veces se itera en los siguientes códigos?
4.2 ¿Cual es el valor final de i en cada caso?

Inciso	Mi resultado	Resultado de la ejecución	Valor final i
a)	4	4	5
b)	2	4	5
c)	4	3	4
d)	cero	cero	cero

Los errores de semántica o sintaxis se corrigieron cambiando los alcances de las variables o declarandolas ”afuera” del controlador, mientras que mi error más común fue contar mal mi resultado con respecto al de la ejecución.

4.3 ¿Cual es el alcance de la variable i, j en cada caso?

a) b) Como solo tenemos la variable ”i” y este se encuentra libre de cualquier llave o paréntesis, su alcance es en todo el código.

c) d) Si tomamos el código con errores del pdf es claro que el alcance tanto de ”i” como de ”j” solo se limita a los paréntesis, aunque si corregimos esto, nuevamente el alcance que van a tener será para todo el código, por lo que serán variables globales.

✪ Preguntas: (Segunda parte)

Responde lo siguiente:

1. ¿Por qué utilizaste esa estructura de control para el manejo del menú?

   Porque si hablamos de opciones o casos, el controlador de flujo ”switch” es el más indicado.

2. Si utilizaste una estructura diferente al switch, ¿Por qué esa implementación es mejor?

   Aunque yo utilicé switch, en algún momento llegué a pensar en usar muchos if's anidados (como una alternativa) pero tras dejarnos usar todos los controladores de flujo vistos en el curso, la opción claramente viable es switch.

3. ¿Cómo puedes mejorar la robustez de tu código o que medidas tomaste para robustecerlo? Aclaro, no estoy pidiendo la implementación de excepciones en java.

   De manera concreta, implenté un mensaje el cual aparece cuando en algunas situaciones el programa no recibe un tipo de dato o valor esperado; lo cual nos lleva a repetir el proceso una y otra vez hasta que los datos sea correctos.