Week 2

1. ¿Recuerda el Totito que realizo en React?, ahora lo estaremos realizando en Java, su programa al iniciar, debe de pedir el nombre de los dos jugadores, después de ingresar el segundo nombre, se debe de elegir de manera aleatoria quien inicia el juego (El juego se inicia con la “X”) . Usted debe de instanciar el tablero y mostrar el tablero en pantalla.

```
 1 | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9
```

   Al inicio el tablero se debe de mostrar como se muestra arriba, pero luego del primer turno, debe de cambiar para mostrar los “X” y “O” según se vayan colocando en el tablero, en cada turno, el programa debe de pedir al usuario ingresar un numero que corresponda a una de las casillas del tablero y colocar el símbolo, las reglas son las mismas del totito, y el programa debería de terminar cuando uno de los dos gane o queden empate, al finalizar siempre mostrar el nombre del ganador.

   ** Su programa debería de tener una clase principal que obtenga el input del usuario, y el tablero debe de estar representado por la clase **Totito** que dentro de ella usted debe de representar el tablero como una matriz. Queda a su discreción el tipo de dato que guarde la matriz.

2. Usted debe de implementar un “To Do List” en Java, aquí encontrara las definiciones de la clase a utilizar, (**Task**, **TaskList**) usted no debe modificar el código que se le entrega, solo debe de completar lo que se le pide (/* Your code here */).

   Clase **Task** (Representación de una tarea)
   - Toda Task debe de tener una fecha de entrega
   - Se debe poder crear un nuevo objeto task con solo un mensaje “msg_reminder”, de ser así, la fecha de entrega sería la fecha de “Hoy” (El día en el que se creó la task)
   - Se debe poder crear un nuevo objeto task con un mensaje “msg_reminder”, una prioridad de Task “priority” y una fecha especificada “do_date”
   - Se debe poder crear un nuevo objeto task con un mensaje “msg_reminder”, una prioridad de Task “priority”, una fecha especificada “do_date” y notas adicionales “notes”
   - Se debe tener un método llamado “getTaskDate” que devuelva en un string la fecha de la task con el siguiente formato: “yyyy-MM-dd” usted puede hacer uso de la variable “pattern” y la clase “SimpleDateFormat”

   Clase **TaskList** (Representación de una lista de tareas)
   - Una lista se puede crear vacía, pero es necesario que tenga un nombre “name”
   - Una lista se puede crear una lista con un nombre “name” y una “Task”
   - Una lista debe de tener la habilidad que, una vez creada la lista, se debe de poder agregar Tasks
   - Se debe poder imprimir la lista completa en consola

** Una vez usted haya completado lo solicitado, es necesario que usted cree una clase que haga uso de las clases TaskList y Task. Presentando cada uno de los puntos solicitados

3. Usted debe de crear su propia implementación de la clase **Stack**, para esta implementación debe de llamar a su clase **MyStack**, recuerde utilizar (*Generic Types*), por lo tanto, su stack debe de soportar cualquier tipo de dato. En cuanto a la implementación, usted es libre de utilizar la estructura que mas encuentre adecuada, con la excepción de la estructura Stack, la cual está implementando usted. Los métodos por implementar de **Stack** son los siguientes:

Methods

Modifier and Type	Method and Description
boolean	`empty()` Tests if this stack is empty.
E	`peek()` Looks at the object at the top of this stack without removing it from the stack.
E	`pop()` Removes the object at the top of this stack and returns that object as the value of this function.
E	`push(E item)` Pushes an item onto the top of this stack.

Adicional a su clase **MyStack**, cree una clase donde hace prueba de todos los metodos implementados.

4. A su instructor no le dio tiempo de terminar con toda la implementación con de la clase **Slist**, la cual representaba una **Singly LinkedList**, por lo tanto, es ahora su tarea de terminar dicha implementación. Los métodos que debe de implementar son los que están en negro, los cuales no fueron implementados.

Methods:

- boolean isEmpty()
- int size()
- void addFirst(E e)
- void addLast(E e)
- E removeFirst()
- E removeLast()
- E remove(int index)
- String toString()

5. Used debe de crear un clon simple de Spotify, para ello debe de crear una clase llamada **Spoty**, el constructor de esta clase debe de recibir un String con el siguiente formato: "nombreCancion1: nombreCancion2: nombreCancion3", estas canciones se deben de agregar a la "base de datos" de su programa, debe de existir un método que inicie la reproducción de las canciones, y otros métodos para poder cambiar de canción en reproducción. Al inicio, ninguna canción se esta reproduciendo.
    - **String getActualSong()** : Devuelve el nombre de la canción que actualmente está reproduciéndose
    - **void next()**: No devuelve nada, pero la siguiente canción en cola se comienza a reproducir, en caso de que se este reproduciendo la ultima canción se pasa a reproducir la primera**.**
    - **void prev()**: No devuelve nada, pero la canción anterior se comienza a reproducir, en caso se este reproduciendo la primera canción, se pasa a reproducir la última canción.
    - **void random()**: No devuelve nada, pero selecciona una canción a reproducir, de forma aleatoria.
    - **void playThisSong(int index)**: No devuelve nada, pero pone en reproducción la canción que corresponda con el índice indicado.
    - **String toString()** : Debe de mostrar la lista de reproducción , mostrando que canción se está reproduciendo actualmente, en el siguiente ejemplo la canción con nombre: nombreCancion3 esta siendo reproducida.
    - ****Para probar su clase, puede hacer uso del siguiente archivo: **SpotyTest.java**

```
1. nombreCancion1
2. nombreCancion2
3. [[ nombreCancion3 ]]
4. nombreCancion4
5. nombreCancion5
6. nombreCancion6
7.
```