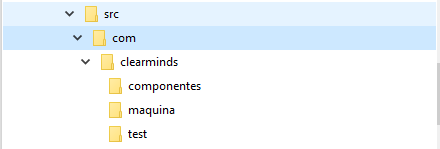
**Ejemplo Evaluación Java Standard Edition - LISTAS**

Crear un nuevo proyecto con el nombre **Evaluacion1**<NombreApellido>, por ejemplo **Evaluacion1SantiagoMosquera.**

En caso de ser necesario puede ocupar las clases del Repaso anterior de Máquina de Dulces, en los primeros ítems, luego ya se usa listas

En su proyecto crear los paquetes  de tal manera que la estructura del proyecto a nivel de carpetas quede de esta manera



En el paquete **componentes** crear la clase que se detalla a continuación:

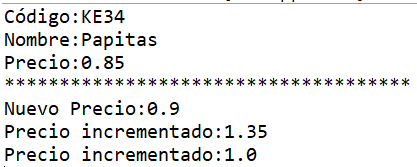
* **Producto:**

|  |  |
| --- | --- |
| Atributos | * **Nombre**: Es el nombre del producto. * **Precio**: Es el precio del producto en números decimales. * **Código**: Es un conjunto de caracteres. |
| Métodos | * Método **incrementarPrecio**: recibe como parámetro un porcentaje de incremento de tipo entero, no tiene retorno.  Calcula el porcentaje de incremento y le suma al precio actual. Ejemplo: si el precio actual es 0.50 y se invoca a incrementarPrecio(10), debe incrementar el 10% al precio actual, es decir el nuevo precio debe ser 0,55. * Método **disminuirPrecio**: recibe un valor de descuento de tipo double, no retorna nada. Resta el valor de descuento al precio actual. |
| Constructores | * Constructor que reciba 3 parámetros correspondientes a los atributos de **Producto** y que asigne sus valores a los atributos. |

# EJERCICIO 1

En el paquete **test** copiar la clase **TestProducto** que se encuentra adjunta en su carpeta de **Recursos**.

Resultado esperado:



En el paquete **componentes** crear la clase que se detalla a continuación:

* **Celda**

|  |  |
| --- | --- |
| Atributos | * **Producto**: de tipo Producto. * **Stock**: de tipo entero * **Código**: Es un conjunto de caracteres. |
| Métodos | * Método **ingresarProducto**: recibe como parámetro  el Producto y el stock inicial, no retorna nada. Colocar los valores en los atributos producto y stock respectivamente. |
| Constructores | * Constructor que recibe 1 parámetro correspondiente al **Código** de la celda y que asigna su valor al atributo correspondiente |

# EJERCICIO 2

En el paquete **test** copiar la clase **TestCelda** que se encuentra adjunta en su carpeta **Recursos.** Ejecutar la clase sin modificarla. Esta ejecución arrojará una excepción del tipo NullPointerException.

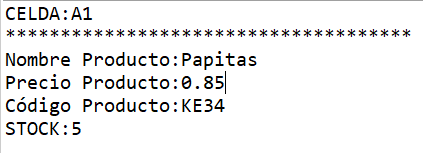
En la clase **TestCelda**, dejar un comentario indicando en qué clase y en qué línea exacta de esa clase da el NullPointerException y además indicar qué variable es la que ocasiona el NullPointer.

# EJERCICIO 3

En el paquete **test**, colocar la clase **TestContenidoCelda,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:

Resultado esperado:



En el paquete **maquina** crear la clase que se detalla a continuación:

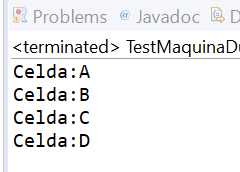
* **Maquina dulces**

|  |  |
| --- | --- |
| Atributos | * **Celdas:** del tipo ArrayList<Celda> * **Saldo:** de tipo double |
| Métodos | * Método **agregarCelda**: recibe el código de la celda, instancia una celda con dicho código y la agrega a la lista * Método **mostrarConfiguracion**: método que no recibe parámetros y no tiene retorno, imprime en consola los códigos de todas las celdas que tenga la máquina en ese momento. |
| Constructores | * Constructor por defecto. |

# EJERCICIO 4

En el paquete **test**, colocar la clase **TestMaquinaDulces,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:



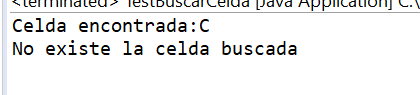
En MaquinaDulces, agregar los siguientes métodos

* Método **buscarCelda:** recibe un código de celda y retorna la Celda correspondiente a dicho código, si no existe retorna null.

# EJERCICIO 5

En el paquete **test**, colocar la clase **TestBuscarCelda,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:

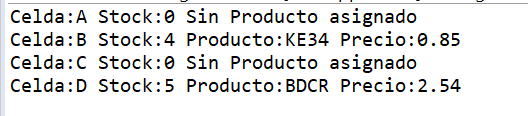


* Método **cargarProducto**: recibe un atributo de tipo Producto, un código de celda y la cantidad de items a colocar inicialmente de dicho producto. Invoca al método **buscarCelda**, guarda el retorno de este método en una variable **celdaRecuperada** de tipo celda. En la celdaRecuperada**,** agrega el producto y el stock inicial usando el método **ingresarProducto** de la celda. Este método no tiene retorno
* Método **mostrarProductos:** Método que no recibe parámetros y no tiene retorno. imprime de cada celda: el código de la celda, el stock actual, el nombre y precio del Producto de la celda.

# EJERCICIO 6

En el paquete **test**, colocar la clase **TestBuscarCelda,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:

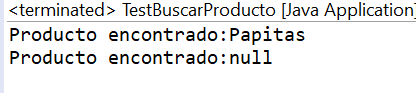


* Método **buscarProductoEnCelda:** recibe el código de la celda y retorna el Producto guardado en esa celda, si no existe la celda, retorna null.

# EJERCICIO 7

En el paquete **test**, colocar la clase **TestBuscarProductoEnCelda,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:

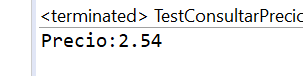


* Método **consultarPrecio:**recibe el código de la celda, retorna el precio del Producto guardado en dicha celda.

# EJERCICIO 8

En el paquete **test**, colocar la clase **TestConsultarPrecio,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:

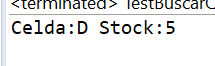


* Método **buscarCeldaProducto:** recibe el código de producto y retorna la celda que lo contiene, si no encuentra retorna null.

# EJERCICIO 9

En el paquete **test**, colocar la clase **TestBuscarCeldaProducto,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:

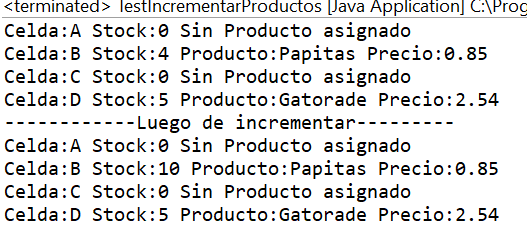


* Método **incrementarProductos:** recibe el código del producto y la cantidad de items a incrementar, no retorna nada. Invoca al método **buscarCeldaProducto** usando el código del producto que recibe. Guarda el retorno de **buscarCeldaProducto** en una variable llamada **celdaEncontrada**.En la celdaEncontrada, agrega el número de items al stock actual de productos en dicha celda.

# EJERCICIO 10

En el paquete **test**, colocar la clase **TestIncrementarProductos,** que se encuentra adjunta en su carpeta de **Recursos.**

Resultado esperado:



* Método **vender:** recibe el código de celda, no retorna nada. Busca la celda y disminuye en 1 el stock. Obtiene el precio del producto y sumar este valor al saldo actual de la máquina.  el método **mostrarProductos** para que también imprima el saldo de la máquina.

# EJERCICIO 11

En el paquete **test**, crear la clase **TestVender,** probar el método vender. Cargar sus propios productos

* Método **venderConCambio:** recibe el código de la celda y el valor ingresado por el cliente en la máquina, retorna el cambio que debe dar la máquina. Busca la celda, resta 1 al stock del producto, modifica el saldo, retorna el vuelto que le debe entregar al cliente.

# EJERCICIO 12

En el paquete **test**, crear la clase **TestVenderConCambio,** probar el método venderConCambio. Cargar sus propios productos

En MaquinaDulces, agregar un nuevo método llamado **buscarMenores**, recibe un double llamado limite y retorna un ArrayList<Producto> con todos los productos cuyo precio es menor o igual que limite.

# EJERCICIO 13

En el paquete **test**, crear la clase **TestBuscarMenores,** agregar 6 celdas, cada celda con un producto diferente. Probar el método buscarMenores, imprimir en el test la lista que retorna el método buscarMenores.