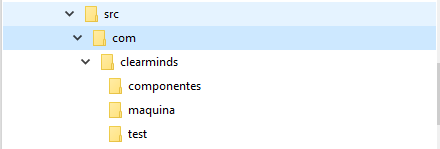
**Repaso Evaluación Java Standard Edition**

Dentro de la carpeta Evidencias crear una subcarpeta con su nombre y apellido y subir la evidencia

En la siguiente evaluación aplicar todo lo revisado durante la capacitación: encapsulamiento, estándares de programación, constructores, métodos, composición de objetos.

1. Crear un nuevo proyecto con el nombre **example\_evaluacion**\_java, por ejemplo **EvaluacionSMO.**
2. En su proyecto crear los paquetes de tal manera que la estructura del proyecto a nivel de carpetas quede de esta manera

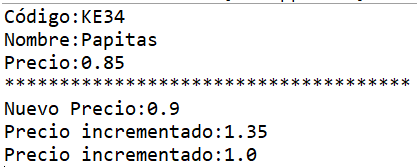


1. En el paquete **componentes** crear la clase que se detalla a continuación:

* **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. |

1. En el paquete **test** copiar la clase **TestProducto** que se encuentra adjunta en su carpeta de **Recursos**. Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



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

* **Celda**

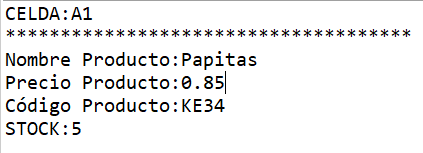
|  |  |
| --- | --- |
| Atributos | * **Producto**: de tipo Producto. * **Stock**: de tipo entero * **Código**: Es un conjunto de caracteres. |
| Métodos | * Método **ingresarProducto**: recibe como parámetro el Producto y el stock inicial, no retorna nada. Colocar los valores en los atributos producto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

1. 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.

1. En el paquete **test**, colocar la clase **TesContenidoCelda,** que se encuentra adjunta en su carpeta de **Recursos.** Agregar lo necesario para que esta compile.

Resultado esperado:

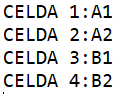


1. En el paquete **maquina** crear la clase que se detalla a continuación:

* **Maquina dulces**

|  |  |
| --- | --- |
| Atributos | * **Celda1**: de tipo Celda. * **Celda2**: de tipo Celda. * **Celda3**: de tipo Celda. * **Celda4**: de tipo Celda. * **Saldo:** de tipo double |
| Métodos | * Método **configurarMaquina**: recibe 4 códigos de celda de tipo String, no retorna nada y los asigna a cada una de las celdas de la máquina * Método **mostrarConfiguracion**: método que no recibe parámetros y no tiene retorno, imprime en consola los códigos de las 4 celdas y el saldo actual de la máquina. * Método **buscarCelda:** recibe un código de celda y retorna la Celda correspondiente a dicho código, si no existe retorna null. * 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. * Método **buscarProductoEnCelda:** recibe el código de la celda y retorna el Producto guardado en esa celda, si no existe la celda, retorna null. * Método **consultarPrecio:**recibe el código de la celda, retorna el precio del Producto guardado en dicha celda. * Método **buscarCeldaProducto:** recibe el código de producto y retorna la celda que lo contiene, si no encuentra retorna null. * 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. * 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. * 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. |
| Constructores | * Constructor por defecto. |

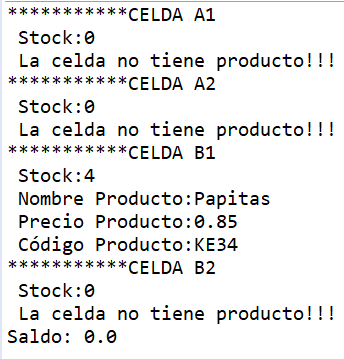
1. En el paquete **test** colocar la clase **TestConfiguracion,**que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado con el formato:



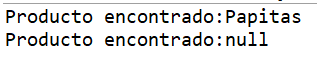
1. En el paquete **test** colocar la clase **TestBusqueda,**que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



1. En el paquete **test** colocar la clase **TestCargarProducto,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



1. En el paquete **test** colocar la clase **TestBuscarProducto,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



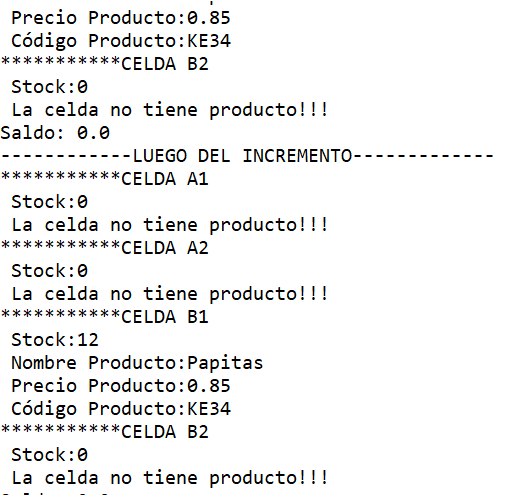
1. En el paquete **test** colocar la clase **TestBuscarProductoNull,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y Agregar un comentario en el código indicando la línea exacta en la clase que dio el NullPointerException y la variable que produjo el error.
2. En el paquete **test** colocar la clase **TestConsultarPrecio,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



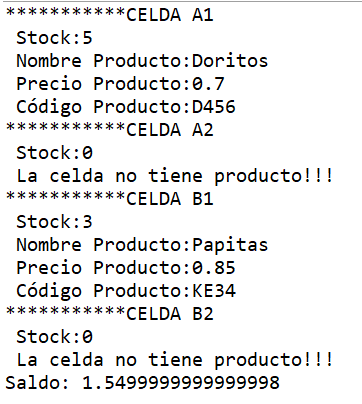
1. En el paquete **test** colocar la clase **TestBuscarCeldaProducto,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



1. En el paquete **test** colocar la clase **TestIncrementarProductos,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



1. En el paquete **test** colocar la clase **TestVender,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:



1. En el paquete **test** colocar la clase **TestVenderConCambio,** que se encuentra adjunta en su carpeta de **Recursos.** Ejecutar la clase sin modificarla y validar que se obtenga el siguiente resultado:

