Entornos de Desarrollo

Tarea 3: Prueba de Compilador en Java

¿Qué es el JDK de Java y el JRE? ¿Qué diferencias hay entre ellos?

JDK o Java Development Kit es el kit de desarrollo de Java, una tecnología de Java desarrollada por Oracle Corporation que implementa la Especificación de Lenguaje de Java (JLS) y la Especificación de Máquina Virtual de Java (JVMS).

Sirve para construir programas usando el lenguaje de programación Java. Trae herramientas útiles como el compilador (javac), el desensamblador de binarios (javap) o el debugger, entre otras herramientas. Provee herramientas de evaluación de rendimiento de aplicaciones, como son VisualVM y Mission Control. Todo esto y más herramientas.

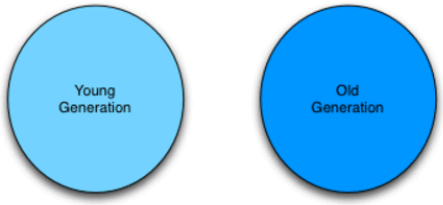
JRE o Java Runtime Environment es el entorno de ejecución de Java. Contiene a la JVM y otras herramientas que permiten la ejecución de las aplicaciones Java. JRE no posee compiladores ni herramientas para desarrollar las aplicaciones Java, solo posee las herramientas para ejecutarlas.

En definitiva, JDK es una herramienta para desarrolladores en Java mientras que JRE está más orientada a los usuarios finales ya que solo permite ejecutar.

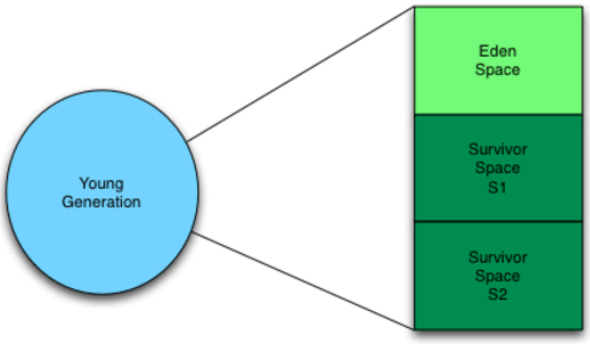
¿Qué es el Java Garbage Collector? ¿Qué hace y cómo funciona?

Es un administrador de memoria que elimina de forma automática los objetos que ya no se encuentran en uso, ahorrándole al desarrollador la tarea de ejecutar un código específico para ello como sería el caso. A continuación citaremos un artículo de la web "arquitecturajava":

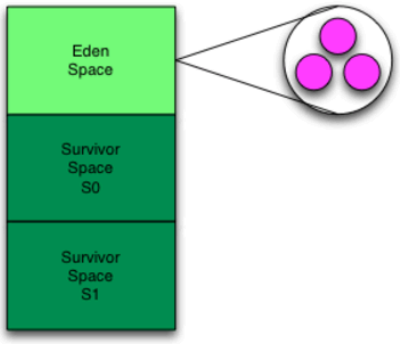
¿Cómo funciona el Java Garbage Collector exactamente?. Java divide la memoria en dos bloques fundamentales , **Young generation y Old generation.**



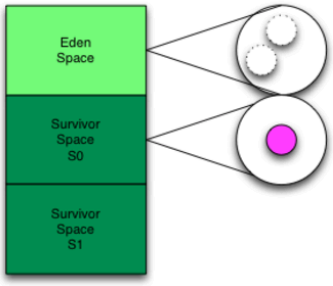
En la zona de Young Generation se almacenan los objetos que se acaban de construir en el programa . Esta zona de memoria se divide en Eden Space ,Survivor Space (S0 y S1)



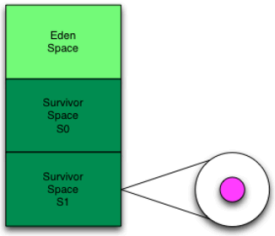
La zona de Eden es la zona en la que los objetos que acabamos de construir se almacenan.



Cuando el recolector de basura pasa , elimina todos los objetos que ya no dispongan de referencias en el Eden Space y los supervivientes los mueve al Survivor Space.



El recolector de basura volverá a pasar otra vez y si queda algún objeto superviviente en el Survivor Space S0 lo moverá al Survivor S1



Después de varias pasadas del recolector de basura los objetos que todavía están vivos en el Survivor Space dejan de ser considerados Young Generation **y pasan a ser considerados Old Generation .**

¿Qué diferencia hay entre INT e INTEGER en Java? ¿Qué se puede hacer en cada caso?

Int es un tipo que almacena un valor binario mientras que Integer es un objeto o clase. Por así decirlo, Integer sería como un contenedor que contiene a Int. El valor de Int se puede cambiar, pero no el de Integer, para lo cual deberíamos crear un nuevo objeto Integer.

¿Qué son las API's de Java? Ponga algún ejemplo aclarativo.

Son un conjunto de clases, interfaces y paquetes predefinidos que ofrece Java para facilitar el desarrollo de aplicaciones. Para ello proporcionan funcionalidades listas para ser usadas, evitando que el desarrollador tenga que escribir código desde cero para tareas comunes como manipular cadenas, manejar archivos, trabajar con colecciones, hacer conexiones de red, o gestionar bases de datos.

Un ejemplo muy común en el caso de Java sería ***java.util*** que contiene utilidades como la clase ***Scanner,*** que permite pedirle al usuario que introduzca algún dato en forma de variables de cualquier tipo, o ***ArrayList***, que permite almacenar variables en una lista dinámica.

```java
import java.util.ArrayList;

public class EjemploAPI {
    public static void main(String[] args) {
        // Crear una lista de nombres usando la API ArrayList
        ArrayList<String> nombres = new ArrayList<>();

        // Agregar elementos a la lista
        nombres.add("Juan");
        nombres.add("Ana");
        nombres.add("Carlos");
```

¿Qué elementos contienen un objeto java? Exponga un ejemplo de ello.

Lo primero que contienen los objetos son las ***clases***. Las clases son, por así decirlo, la estructura general o plano dentro del cual se encuentran todos los demás elementos, algo así como una gran muñeca rusa.

Los elementos más comunes dentro de una clase son los ***atributos***, variables que se declaran en privado para que apliquen a todos los métodos de la clase. El conjunto sus valores se denomina ***estado***.

Por ejemplo, en una clase “Coche” un atributo puede ser Private String Marca y otro puede ser Private Int Precio. Son la unidad fundamental sin la cual no sería posible crear un objeto.

Los ***métodos*** son las acciones que la clase ejecuta utilizando los atributos, definidas como ***comportamiento.*** Un tipo de método especial es el ***constructor***, que inicializa los atributos de la clase que van a ser utilizados más adelante por los demás métodos. Para que la ejecución de dichas acciones pueda ser llevada a cabo es necesario una clase principal o ***main***.

Siguiendo una estructura de muñeca rusa, Objeto > Clase > Constructor/Método > atributos.

```java
public class Coche {
    // Atributos
    private String marca;
    private String color;
    private int velocidad;

    // Constructor
    public Coche(String marca, String color) {
        this.marca = marca;
        this.color = color;
        this.velocidad = 0; // Velocidad inicial es 0
    }

    // Método para arrancar el coche
    public void arrancar() {
        System.out.println("El coche ha arrancado.");
    }

    // Método para acelerar
    public void acelerar(int incremento) {
        velocidad += incremento;
        System.out.println("El coche ha acelerado a " + velocidad + " km/h.");
```

¿Qué es el bytecode en Java?

Es el tipo de instrucciones que la máquina virtual Java (JVM) espera recibir para posteriormente ser compiladas a lenguaje de máquina mediante un compilador JIT a la hora de su ejecución. Usualmente es el resultado de utilizar un compilador del lenguaje de programación Java (como javac), pero puede ser generado desde otros compiladores.

¿Qué es la variable de entorno PATH? ¿Qué es la variable de entorno CLASSPATH? ¿Cuáles son sus diferencias?

El entorno PATH es una lista de rutas de directorios que el sistema operativo utiliza para buscar programas ejecutables. Los directorios y comandos están especificados en PATH para poder ejecutar los comandos desde cualquier parte sin necesidad de escribir la ruta completa. Por ejemplo:

Sin PATH configurado:

```
C:\Program Files\Java\jdk-17\bin\javac HolaMundo.java
```

Con PATH configurado:

```
javac HolaMundo.java
```

Classpath es un parámetro de la máquina virtual Java o del compilador Java que especifica la ubicación de las clases y los paquetes definidos por el usuario . El parámetro se puede configurar en la línea de comandos o mediante una variable de entorno.

Diferencias: PATH busca programas ejecutables, CLASSPATH busca clases y archivos .jar. Path afecta a comandos del sistema y CLASSPATH afecta a la ejecución de programas en Java. Los contenidos dentro de PATH son directorios que contienen ejecutables, los de CLASSPATH son directorios y archivos que contienen clases. Además los programas de CLASSPATH necesitan librerías externas, mientras que en PATH javac y java están disponibles desde cualquier lugar.

En esta práctica se va a utilizar el compilador de Java en la línea de comandos de Windows (cmd) conocido como javac que creará un fichero objeto nombre_fichero.class que podrá ser ejecutado con la JVM.

La práctica se trata de realizar una clase con las siguientes características:

- Solo tenga un atributo de tipo String de un solo carácter que se llame ***simboloDibujado***.

- Un método constructor de clase que asigne como carácter inicial el '*'.

- Un método que pueda cambiar el símbolo de dibujado llamado ***cambiarSimbolo***(String nuevo).

- Un método que me dibuje con dicho símbolo un cuadrado relleno de lado un número entero mayor o igual que 1, llamado cuadrado(int n).

- Un método que dibuje un rectángulo relleno de lados a y b llamado rectangulo(int a, int b) con a y b mayor o igual que 1.

- Un método que dibuje un cuadrado de lado n pero hueco.

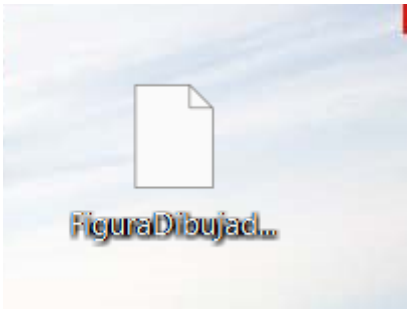
- Otro método llamado rectanguloHueco(int a , int b) que dibuje un rectángulo hueco de lados a y b, con a y b mayores o iguales que 1.

- Un método que dibuje un triángulo rectángulo a la izquierda relleno llamado ***trianguloRectIzq***(int h) donde h es la altura con valor de 2 en adelante.

- Un método que dibuje un triángulo rectángulo a la derecha relleno llamado ***trianguloRectDer***(int h) donde h es la altura con valor de 2 en adelante.

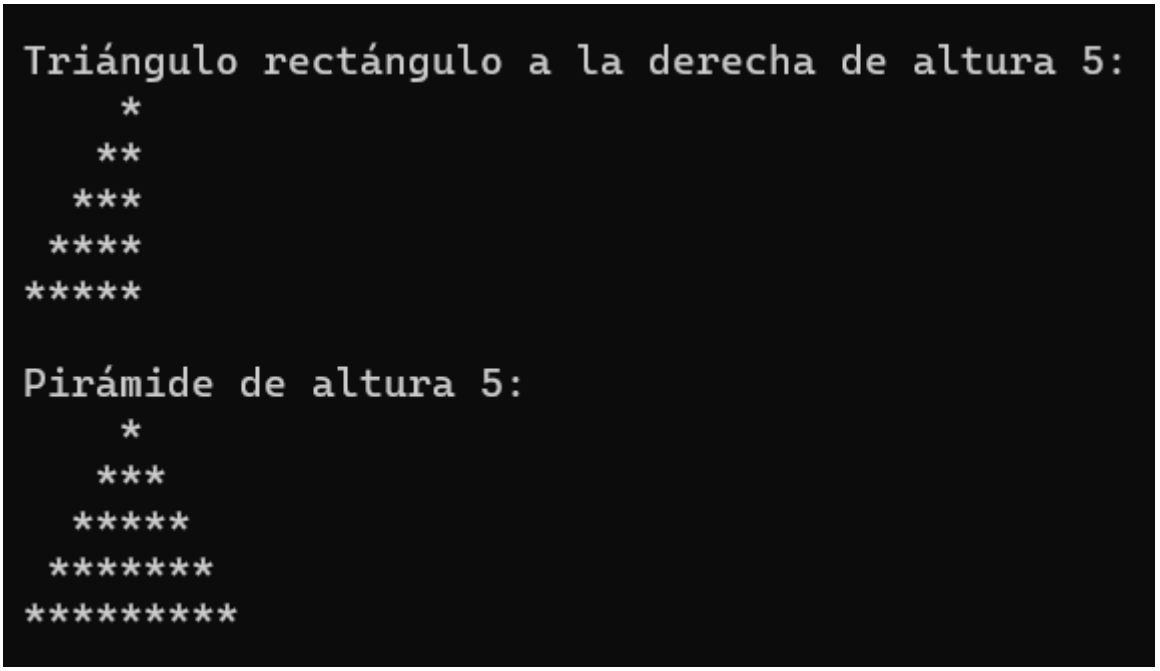
- Por último un método que dibuje una pirámide llamado ***piramide***(int h), donde h es la altura de la pirámide.

Primero creamos el archivo con el código en el blog de notas y se guarda en formato .java:



Después se ejecuta el archivo desde la consola cmd:

```
C:\Users\guill\Desktop>javac FiguraDibujada.java

C:\Users\guill\Desktop>java FiguraDibujada
Cuadrado relleno de lado 5:
*****
*****
*****
*****
*****

Rectángulo relleno de 4x6:
******
******
******
******

Cuadrado hueco de lado 5:
*****
*   *
*   *
*   *
*****

Rectángulo hueco de 4x6:
******
*    *
*    *
******

Triángulo rectángulo a la izquierda de altura 5:
*
**
***
****
*****
```



A continuación mostraré el código empleado:

```
public class FiguraDibujada {

    // Atributo
    private String simboloDibujado;

    // Constructor que asigna el símbolo inicial como '*'
    public FiguraDibujada() {
        this.simboloDibujado = "*";
    }

    // Método para cambiar el símbolo de dibujado
    public void cambiarSimbolo(String nuevo) {
        if (nuevo.length() == 1) {
            this.simboloDibujado = nuevo;
        } else {
            System.out.println("Error: El símbolo debe ser un solo carácter.");
        }
    }

    // Método para dibujar un cuadrado relleno
    public void cuadrado(int n) {
        if (n >= 1) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    System.out.print(this.simboloDibujado);
                }
                System.out.println();
            }
        } else {
            System.out.println("Error: El tamaño del cuadrado debe ser mayor o igual a 1.");
        }
    }

    // Método para dibujar un rectángulo relleno
    public void rectangulo(int a, int b) {
        if (a >= 1 && b >= 1) {
            for (int i = 0; i < a; i++) {
                for (int j = 0; j < b; j++) {
                    System.out.print(this.simboloDibujado);
                }
                System.out.println();
            }
        } else {
            System.out.println("Error: Los lados del rectángulo deben ser mayores o iguales a 1.");
        }
    }

    // Método para dibujar un cuadrado hueco
    public void cuadradoHueco(int n) {
        if (n >= 1) {
            for (int i = 0; i < n; i++) {
                for (int j = 0; j < n; j++) {
                    if (i == 0 || i == n - 1 || j == 0 || j == n - 1) {
                        System.out.print(this.simboloDibujado);
                    } else {
                        System.out.print(" ");
                    }
                }
                System.out.println();
            }
        } else {
            System.out.println("Error: El tamaño del cuadrado debe ser mayor o igual a 1.");
        }
```

```
}

// Método para dibujar un rectángulo hueco
public void rectanguloHueco(int a, int b) {
    if (a >= 1 && b >= 1) {
        for (int i = 0; i < a; i++) {
            for (int j = 0; j < b; j++) {
                if (i == 0 || i == a - 1 || j == 0 || j == b - 1) {
                    System.out.print(this.simboloDibujado);
                } else {
                    System.out.print(" ");
                }
            }
            System.out.println();
        }
    } else {
        System.out.println("Error: Los lados del rectángulo deben ser mayores o iguales a 1.");
    }
}

// Método para dibujar un triángulo rectángulo a la izquierda
public void trianguloRectIzq(int h) {
    if (h >= 2) {
        for (int i = 1; i <= h; i++) {
            for (int j = 0; j < i; j++) {
                System.out.print(this.simboloDibujado);
            }
            System.out.println();
        }
    } else {
        System.out.println("Error: La altura del triángulo debe ser mayor o igual a 2.");
    }
}

// Método para dibujar un triángulo rectángulo a la derecha
public void trianguloRectDer(int h) {
    if (h >= 2) {
        for (int i = 1; i <= h; i++) {
            for (int j = 0; j < h - i; j++) {
                System.out.print(" ");
            }
            for (int j = 0; j < i; j++) {
                System.out.print(this.simboloDibujado);
            }
            System.out.println();
        }
    } else {
        System.out.println("Error: La altura del triángulo debe ser mayor o igual a 2.");
    }
}

// Método para dibujar una pirámide
public void piramide(int h) {
    if (h >= 2) {
        for (int i = 1; i <= h; i++) {
            for (int j = 0; j < h - i; j++) {
                System.out.print(" ");
            }
            for (int j = 0; j < (2 * i - 1); j++) {
                System.out.print(this.simboloDibujado);
            }
            System.out.println();
        }
    } else {
        System.out.println("Error: La altura de la pirámide debe ser mayor o igual a 2.");
    }
```

```
    }

    // Método principal para probar las formas
    public static void main(String[] args) {
        FiguraDibujada figura = new FiguraDibujada();

        System.out.println("Cuadrado relleno de lado 5:");
        figura.cuadrado(5);

        System.out.println("\nRectángulo relleno de 4x6:");
        figura.rectangulo(4, 6);

        System.out.println("\nCuadrado hueco de lado 5:");
        figura.cuadradoHueco(5);

        System.out.println("\nRectángulo hueco de 4x6:");
        figura.rectanguloHueco(4, 6);

        System.out.println("\nTriángulo rectángulo a la izquierda de altura 5:");
        figura.trianguloRectIzq(5);

        System.out.println("\nTriángulo rectángulo a la derecha de altura 5:");
        figura.trianguloRectDer(5);

        System.out.println("\nPirámide de altura 5:");
        figura.piramide(5);
    }
}
```