Práctico 1

1) Instale el IDE IntelliJ
2) Las tres componentes de la plataforma Java son:

- Java ME (Java Micro Edition):

  Esta parte de la plataforma Java está diseñada para dispositivos con recursos limitados, como teléfonos móviles, PDAs (Asistentes Digitales Personales) y otros dispositivos embebidos.

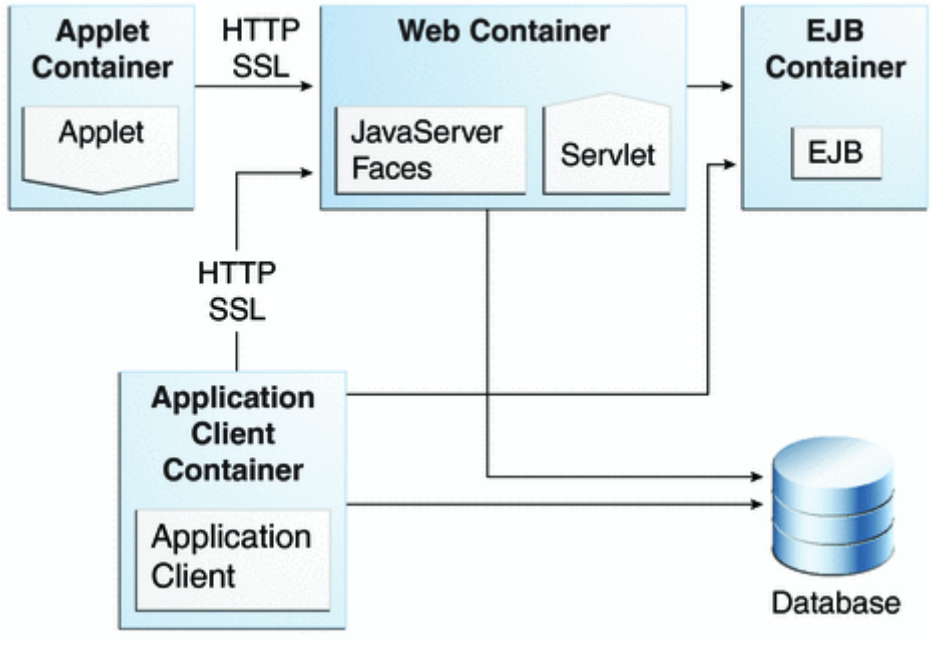
- Java Se (Java Standard Edition):

  Es la parte fundamental de la plataforma Java, proporcionando las APIs (Interfaces de Programación de Aplicaciones) y la biblioteca de clases base para el desarrollo de aplicaciones Java de propósito general. Java SE incluye todo lo necesario para desarrollar y ejecutar aplicaciones Java en una variedad de dispositivos y sistemas operativos. Incluye las bibliotecas básicas, el compilador Java, la máquina virtual Java (JVM), y otras herramientas esenciales para desarrolladores.

https://github.com/santozzi/Proyecto3-TDP-DrApocalipsis

- Java EE (Java Enterprise Edition):

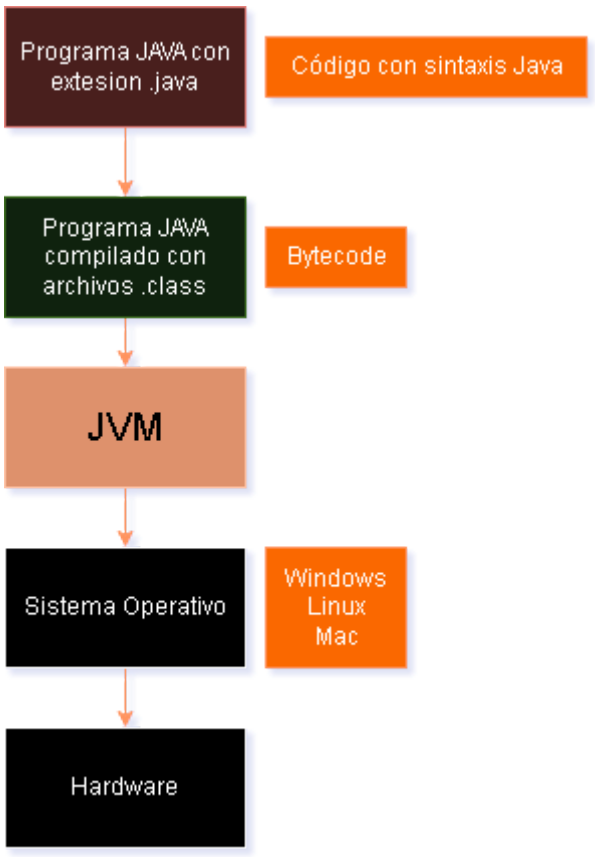
  Anteriormente conocida como J2EE (Java 2 Platform, Enterprise Edition), esta es la plataforma Java diseñada para el desarrollo y ejecución de aplicaciones empresariales. Java EE proporciona un conjunto de especificaciones y APIs adicionales específicamente orientadas a la construcción de aplicaciones de nivel empresarial, como aplicaciones web, aplicaciones distribuidas y servicios web. Incluye funcionalidades como Servlets, JSP (JavaServer Pages), JPA (Java Persistence API), EJB (Enterprise JavaBeans), JMS (Java Message Service), entre otros.



  Actualmente para esta tarea se utilizan frameworks como Spring.

3)La diferencia entre JDK y JRE es que el primero esta pensado para el desarrollador, ya que trae las herramientas para desarrollar, compilar, debuggear y ejecutar un programa java, mientras que el segundo es un runtime, que sirve para ejecutar las aplicaciones Java.

4) JVM: Cuando un desarrollador escribe código lo realiza dentro de archivos con extensión .java, al compilar el programa este se compila en archivos con extensión .class, los cuales estan codificados con un lenguaje llamado bytecode el cual es interpretado por la JVM (Java Virtual Machine), esta estrategia permite escribir el código una vez, y usarlo en diferentes plataformas, mistras estas tengan instalada la JVM.



5)Los tipos de dato primitivo en Java son aquellos datos que no son objetos y no tienen métodos. Estos son:

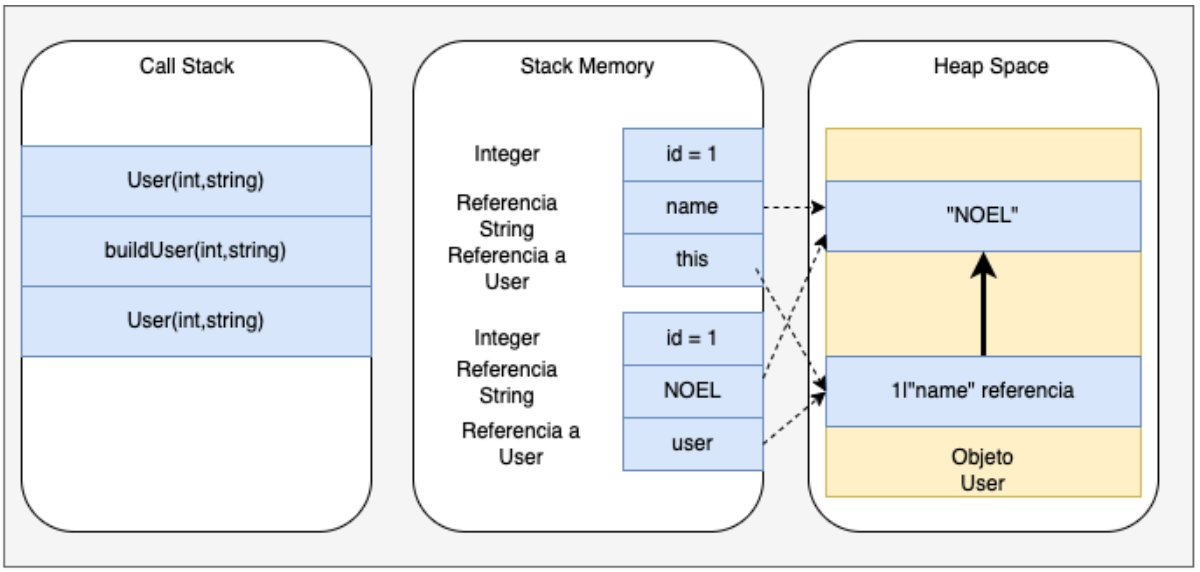
- byte
- short
- int
- long
- boolean
- char
- float
- double

6) Las **variables locales** son variables declaradas dentro de un método, constructor o bloque, y solo existen dentro de ese método, constructor o bloque en particular (variables locales).
Estas nacen cuando se crean y mueren una vez finalizado el bloque, metodo o constructor. Sin embargo las **variables de instancia** son variables declaradas dentro de una clase pero fuera de cualquier método, constructor o bloque, (variables globales). Cada instancia (objeto) de esa clase tiene su propia copia de estas variables. Nacen cuando se instancia una clase y mueren cuando esta instancia es borrada por el “Recolector de basura”.

7)Heap memory es el espacio en donde se guardan los objetos, y Stack memory es en donde se guardan las referencias que apuntan a esos objetos.
Supongamos que tenemos la clase Persona, al escribir:

```
Persona pepe;
```

la variable pepe de tipo Persona se guarda como referencia en el Stack memory y cundo se escribe:
pepe = new Persona()
el objeto {“pepe”, “perez”} se guarda en la Heap memory y es apuntado por la referencia pepe que se encuentra en el Stack memory.



8)Son modificadores de acceso que aportan a un nivel de visibilidad, existen cuatro tipos distintos de modificadores de acceso, algunos aplicables a clases, métodos y variables y otros sólo a métodos y variables;
Estos son:
public, private, protected, default

Modificador	Clase	Package	Subclase	Otros
public	✔	✔	✔	✔
protected	✔	✔	✔	●
default	✔	✔	●	●
private	✔	●	●	●

9)Modificadores que no son de acceso:

- final : Los metodos y los atributos no pueden ser modificados o sobrescrito.

static field → una única copia de ese atributo es creada y compartida por todas las instancias de esa clase. Se almacenan en el heap memory

  - Cuando el valor de la variable es independiente de los objetos.

○ Cuando se supone que se debe compartir el valor en todos los objetos.

● static method → Los métodos estáticos también pertenecen a una clase en lugar del objeto. Podemos llamarlos sin crear el objeto.
○Para acceder a / manipular variables estáticas y otros métodos estáticos que no dependen de los objetos.
Un patrón de diseño donde se explota este modificador es el Singleton, que usa un método estático para crear una instancia si no fue creada o, de lo contrario, devuelve la instancia creada anteriormente, logrando así una clase que puede tener solo una instancia.
● static block → Si las variables estáticas requieren una lógica adicional de declaración múltiple durante la inicialización, podemos usar un bloque estático.
● static class → clases anidadas. dos tipos
○ Las clases anidadas que declaramos static se llaman clases estáticas anidadas.
○ Las clases anidadas que no son estáticas se llaman clases internas.
● Las clases internas tienen acceso a todos los atributos de la clase exterior (incluyendo atributos privados), mientras que las clases estáticas anidadas solo tienen acceso a atributos estáticos de la clase exterior.

- abstract
  Solo puede ser usado en clases abstractas, estas clases tienen la particularidad de que no pueden instanciarse y que pueden tener métodos abstractos, los mismos tienen solo el encabezado del método y las clases hijas estan obligadas a implementarlas.

- transient
  Utilizado para indicar que los atributos y métodos de un objeto no son parte persistente del objeto o bien que estos no deben guardarse y restaurarse utilizando el mecanismo de serialización estándar.

- synchronized
  Los métodos solo pueden ser accesados por un único hilo a la vez.

- volatile
  Se utiliza este modificador sobre los atributos de los objetos para asegurar que el valor siempre está actualizado, a pesar de ser utilizado por varios hilos de ejecución.

- native
  Se invocará a un método nativo vía JNI

10)En Java, los paquetes son estructuras que organizan clases y/o interfaces relacionadas. Se definen con package nombre; en los archivos. Ayudan a evitar conflictos de nombres, controlar el acceso y organizar proyectos. Para usar clases de otros paquetes, se importan con import paquete.Clase;. Las clases del mismo paquete se usan sin importación.
Son muy útiles para modularizar el código, en la arquitectura MVC, se utilizan paquetes para dividir el proyecto en controladores, modelos y vistas.

11)Una variable local en Java se define dentro de un método, constructor o bloque. Su alcance está limitado a ese contexto, no siendo accesible fuera. Se declara con tipo nombreVariable; o tipo nombreVariable = valor;, siendo temporal y específica a su bloque de código.

Este tipo de variables, se crean cuando se inicia el método y se borran al terminar la ejecución del mismo.

12)El método main() en Java es el punto de entrada para ejecutar un programa. Es donde comienza la ejecución y desde donde se llaman otros métodos. Se declara como public static void main(String[] args) y recibe argumentos de la línea de comandos, siendo esencial para iniciar programas Java.

13)En Java, los constructores son métodos especiales, deben tener como nombre el mismo nombre de la clase, que sirven para instanciar una clase e inicializar atributos de la misma. Se llaman automáticamente al crear un objeto con new Clase() y pueden recibir parámetros para configurar el objeto. Pueden sobrecargarse con diferentes cantidad y/o tipo de parámetros. Una misma clase puede tener uno o varios constructores.

14)En Java, una clase abstracta puede tener métodos implementados y atributos, además de métodos abstractos. Las interfaces solo pueden tener encabezados de métodos y constantes. Las clases se extienden en clases hijas una sola vez, pero se pueden implementar múltiples interfaces, lo que permite una mayor flexibilidad en la estructura del código.

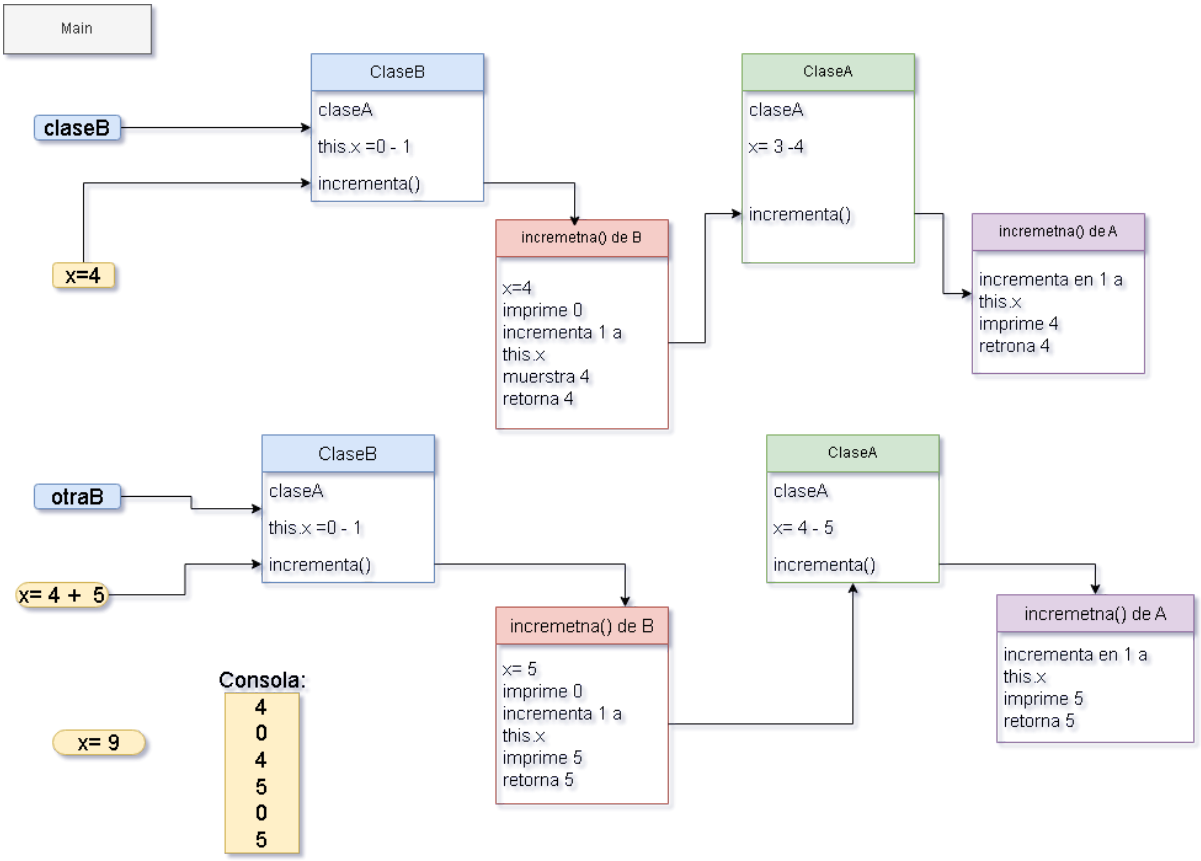
17)No me parece correcto ya que la ClaseA no respeta el principio de encapsulamiento. El estado deberia tener una visibilidad private o protected, y acceder y/o modificar con su getter o su setter.

18)La opcion correcta en la c;
private static final int i = 99;
convierte a i en una constante global la cual no se puede modificar, pero el bloque for crea su propia variable local i al hacer int i. Luego en la siguiente linea, imprime 0 en la siguiente incrementa en 1 a la variable local i y sale del bucle, luego imprime la constante global 99, quedando asi 099.

19) final, que convierte al valor en una constante.

20)La respuesta correcta es la c, ya que es un método abstracto y el mismo no lleva cuerpo solo en encabezado y solo se pueden poner en una clase abstracta.

21)Traza



No lo verifiqué 🙂

15,16,22-26) en github
https://github.com/santozzi/laboratirolII/tree/master