* JDK:
* JRE:
* JEE:
* Patrones de diseño
* Estándares de nombres o nomenclaturas
* Organización de paquetes
* Programación funcional ventajas, desventajas
* Cuándo definir un var o no
* Manejo de excepciones y derivados
* Diferencias entre Java 7, 8 y 9 (Principalmente 8 y sus diferencias con ventajas)
* Tipos de arreglos (Collections) (Ventajas y cuando usar uno u otro)
* Iterar sobre arreglos
* Programación en paralelo (hilos al final)
* Programación asíncrona (al final)

**JDK.**

Java Standard Edition (Java SE) Development Kit (JDK). The JDK is a development environment for building applications, applets, and components using the Java programming language.

The JDK includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform.

**JRE.**

The Java Runtime Environment (JRE) allows to run application written in the Java programming language. Like the JDK, it contains the Java Virtual Machine (JVM), classes comprising the Java platform API, and supporting files. Unlike the JDK, it does not contain development tools such as compilers and debuggers.

You can freely redistribute the JRE with your application, according to the terms of the JRE license. Once you have developed your application using the JDK, you can ship it with the JRE so your end-users will have a Java platform on which to run your software.

**DESIGN PATTERNS**

Are typical solutions to commonly occurring problems in software design. They are like pre-made blueprints that you can customize to solve a recurring design problem in your code.

You can’t just find a pattern and copy it into your program, that way you can with off-the shelf functions or libraries. The pattern is not a specific piece of code, but a general concept for solving a particular problem. You can follow the pattern details and implement a solution that suits the realities of your program.

Patterns are often confused with algorithms, because both concepts describe typical solutions to some known problems. While an algorithm always defines a clear set of actions that can achieve some goal, a pattern is more a high-level description of a solution. The code of the same pattern applied to different programs may be different.

An analogy to an algorithm is a cooking recipe: both have clear steps to achieve a goal. On the hand, a pattern is more like a blueprint: you can see what the result and its features are, but the exact order of implementation is up to you.

Here are the sections that are usually present in a pattern description:

* **Intent** of the briefly describes both the problem and the solution.
* **Motivation** further explains the problem and the solution the pattern makes possible.
* **Structure** of classes shows each part of the pattern and how they are related
* **Code** example in one of the popular programming languages makes it easier to grasp the idea behind the pattern

Some pattern catalogs list other useful details, such as applicability of the pattern, implementation steps and relations with other patterns.

**Why should I learn patterns?**

The truth is that you might manage to work as a programmer for many years without knowing about a single pattern.