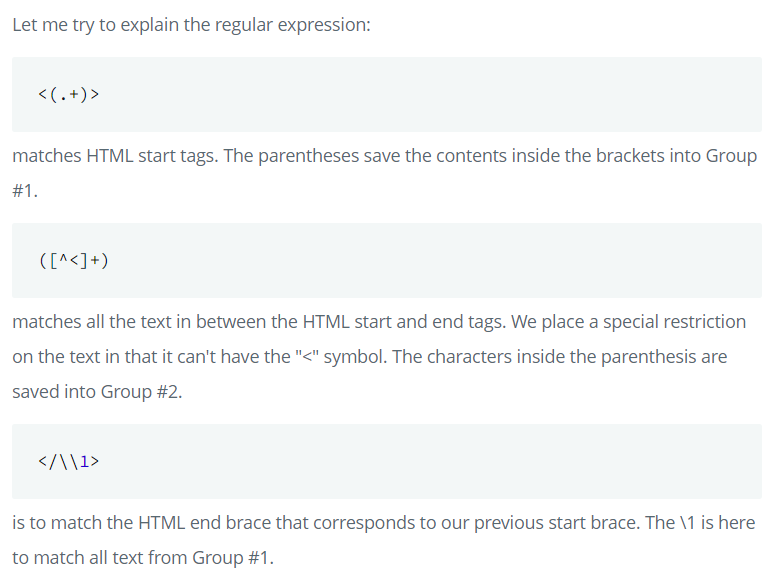
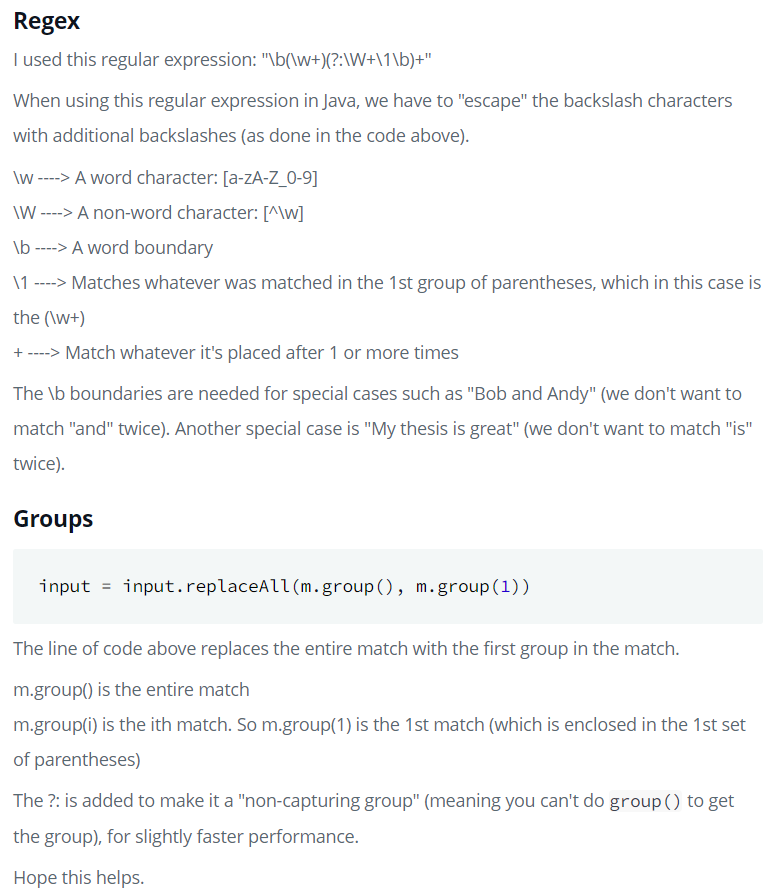
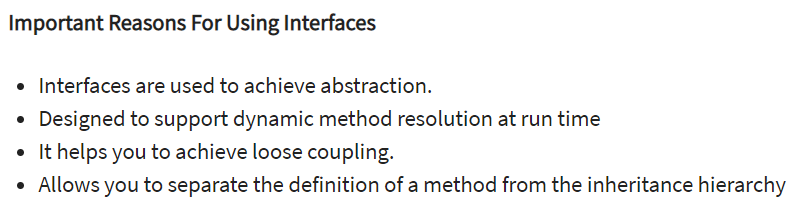
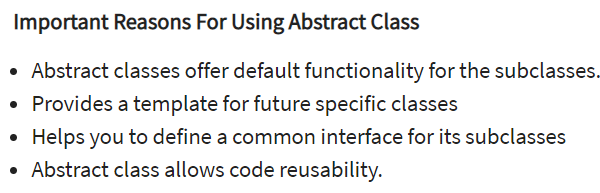
* **Static** is a non-access modifier used for methods and attributes. Static methods/attributes can be accessed without creating an object of a class. Attributes and methods belong to the class, rather than an object.
* **concat()** can be used to append strings to an existing string.
* A ***static initialization block*** is a normal block of code enclosed in braces and preceded by the *static* keyword. A class can have any number of static initialization blocks. The runtime system guarantees that the static initialization blocks are called in the order that they appear in the source code. Private static methods can be reused later if the class variable needs to be reinitialized.
* ***getCurrencyInstance()*** method from the class *NumberFormat* can be used to format Currency data.
* ***length()*** is a method to retrieve the length of a String.
* ***substring(start, end)*** retrieves a substring from a String.



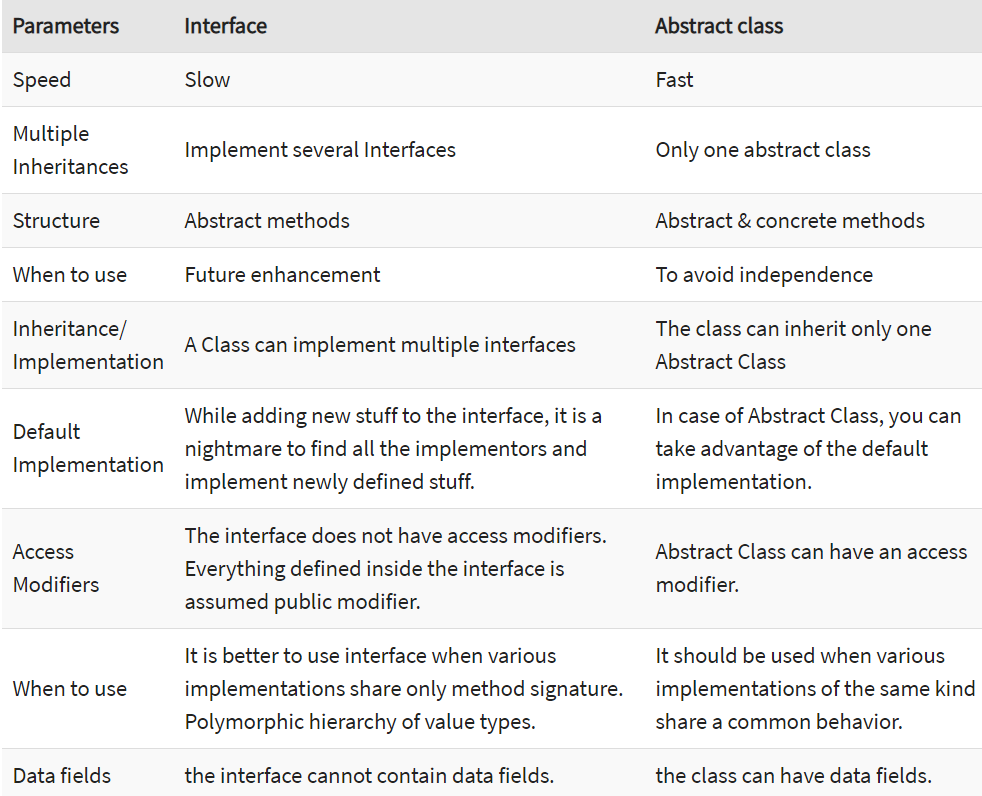


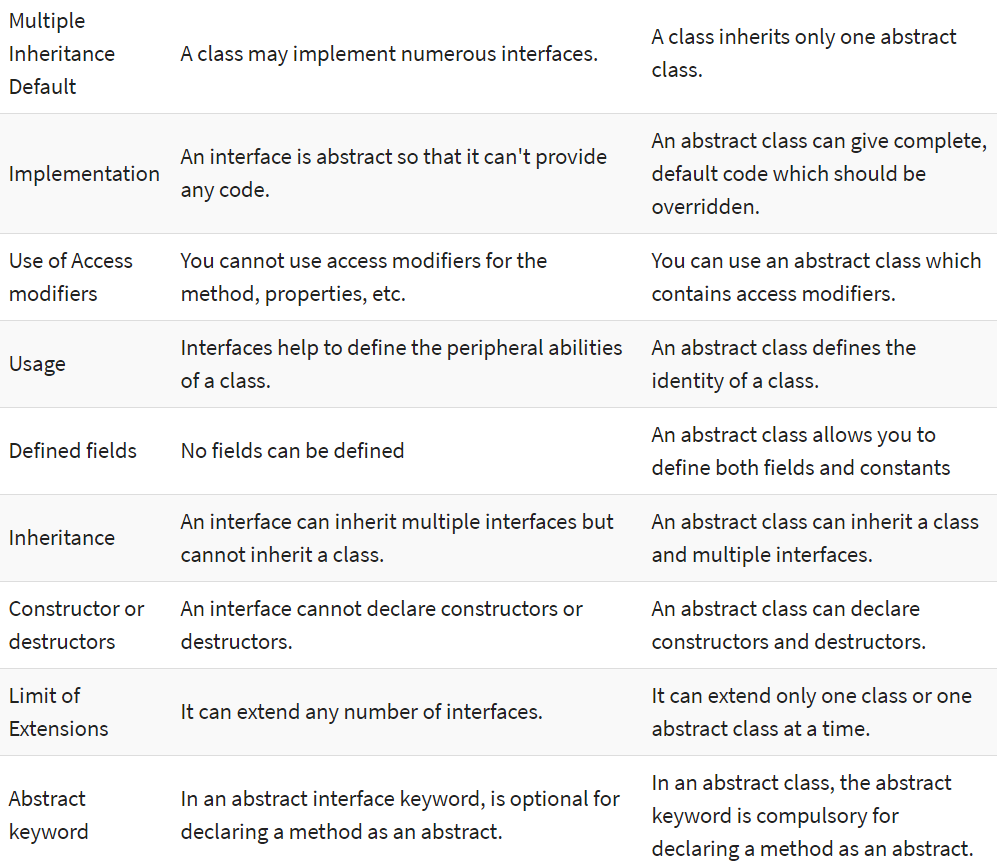
* **Stacks** are collections of elements, with two principal operations: *push*, which adds an element to the collection, and *pop*, which removes the last element that was added. *peek* is a method that retrieves the last element of the Stack without removing the element from it. They work in a LIFO (last in, first out) manner.
* A **Set** is a data type that stores values without any particular order, and no repeated values.
* **Generics** allow to handle multiple datatypes using single methods in an efficient way.
* A **Comparator** interface is used to order the objects of user-defined classes. A Comparator object is capable of comparing two objects of two different classes.
* **BitSet** creates a special type of array that holds bit values. It can increase in size as needed.
* An **interface** is a blueprint that can be used to implement a class. It doesn’t contain any concrete methods, and all of their methods are *abstract*. They never contain instance variables but can contain public static final variables.
* An **abstract class** should have at least one abstract method but can have multiple concrete methods. They allow to create blueprints for concrete classes.

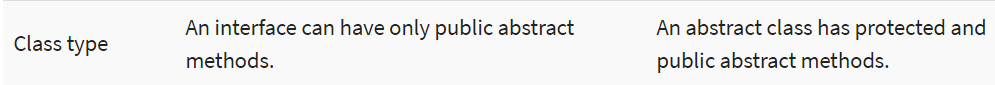


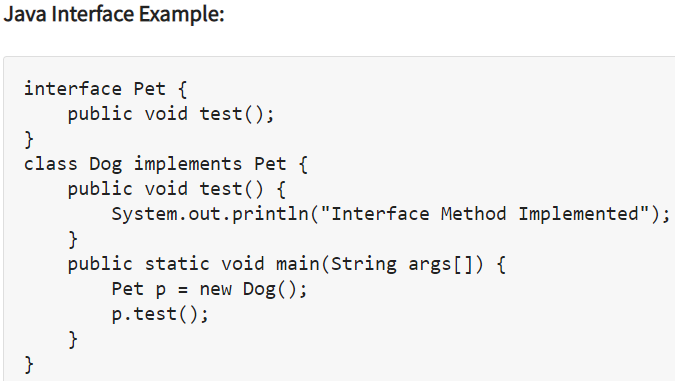


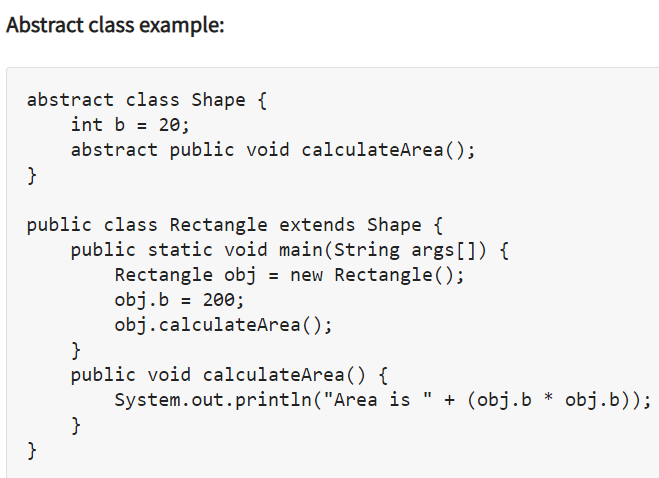
* Abstract classes allow to make functionalities that subclasses can implement or override, while an interface only permits to state functionality but not to implement it. A class can extend only one abstract class, while it can implement multiple interfaces.

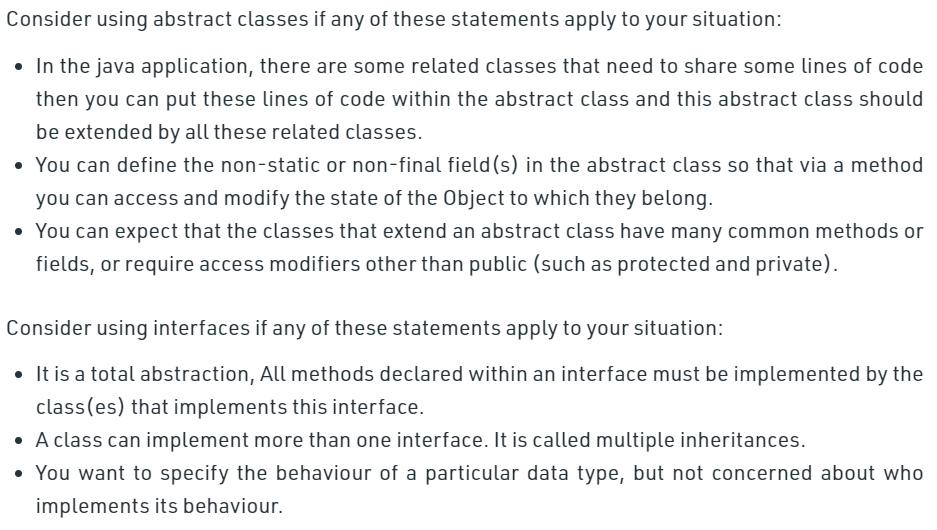












* **Inheritance** is the process where one class acquires the properties (methods and fields) of another. *extends* is the keyword used to inherit properties of a class.