* **Object:** An object stores its state in *fields* (variables in some programming languages) and exposes its behavior through *methods* (functions in some programming languages).
* Hiding internal state and requiring all interaction to be performed through an object's methods is known as **data encapsulation.**
* A **class** is the blueprint from which individual objects are created.
* Each class is allowed to have one direct superclass, and each superclass has the potential for an unlimited number of subclasses, Inheritance is done using the keyword **extends.**
* **Interfaces** is a group of related methods with empty bodies. They can be implemented using the keyword **implements.** Any number of interfaces can be implemented using a comma after the previous one in the code.
* **Primitive Data Types:**
  + Byte – 8 bits, default value – 0
  + Short – 16 bits, default value – 0
  + Int – 32 bits, default value - 0
  + Long – 64 bits, default value - 0
  + Float – 32 bits, default value – 0.0
  + Double – 64 bits, default value – 0.0
  + Boolean – 1 bit, default value - false
  + Char – 16 bits, default value - '\u0000'
* Operators
  + +, -, \*, /, %
  + == equal to
  + != not equal to
  + > greater than
  + >= greater than or equal to
  + < less than
  + <= less than or equal to
  + && Conditional-AND
  + || Conditional-OR
  + The bitwise & operator performs a bitwise AND operation.
  + The bitwise ^ operator performs a bitwise exclusive OR operation.
  + The bitwise | operator performs a bitwise inclusive OR operation.
* **Control Flow Statements:**
  + if-then
  + if-then-else
  + while, do-while
* **Annotations-** a form of metadata, provide data about a program that is not part of the program itself. Its Uses are
  + Information for the compiler
  + Compile-time and deployment-time processing
  + Runtime processing
  + Examples - @Override, @Author…
* If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**. Method overloading is done within the same class
* If subclass (child class) has the same method as declared in the parent class, it is known as **method overriding in java**. Method overriding is done among different classes.
* All of the numeric **wrapper classes** are subclasses of the **abstract class Number**:
  + Byte, Short, Integer, Long, Float, Double
* Using Math class Example: Math.PI   
   But if we use import static java.lang.Math.\*;

Then we can invoke the Math class methods by their simple names

Example: abs(a);

* Math.random() method returns a pseudo-randomly selected number between 0.0 and 1.0.  
   example: to generate an integer between 0 and 9,  
   int number = (int)(Math.random() \* 10);
* Using toString(), will convert its primitive type to a string

Example: String s = Integer.toString(i);

* String class also have methods like split(), toLowerCase(), toUpperCase(), and valueOf().
* Autoboxing is the automatic conversion that the Java compiler makes between the primitive types and their corresponding object wrapper classes. For example, converting an int to an Integer, a double to a Double, and so on. If the conversion goes the other way, this is called unboxing.
* An **exception** is an event, which occurs during the execution of a program, that disrupts the normal flow of the program's instructions.
  + When an error occurs within a method, the method creates an object and hands it off to the runtime system.
* **Exception Handling** is done using try-catch block. It supports multiple catch blocks.

Syntax: try{

//code that may throw exception

}catch(Exception\_class\_Name ref){}

IMPORTANT: An exception handler is considered appropriate if the type of the exception object thrown matches the type that can be handled by the handler.

* The **finally** block always executes when the try block exits whether exception is handled or not.
* **Throw** keyword used to explicitly throw an exception mainly customized exceptions