Java Concepts

**Class :** A class is a user defined blueprint or prototype from which objects are created.

1. **Modifiers**: A class can be public or has default access (Refer [this](https://www.geeksforgeeks.org/access-specifiers-for-classes-or-interfaces-in-java/) for details).
2. **class keyword:**class keyword is used to create a class.
3. **Class name:** The name should begin with an initial letter (capitalized by convention).
4. **Superclass(if any):** The name of the class’s parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
5. **Interfaces(if any):** A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
6. **Body:** The class body surrounded by braces, { }.

**Object :** It is a basic unit of Object-Oriented Programming and represents the real life entities.

1. **State**: It is represented by attributes of an object. It also reflects the properties of an object.
2. **Behavior**: It is represented by methods of an object. It also reflects the response of an object with other objects.
3. **Identity**: It gives a unique name to an object and enables one object to interact with other objects.

Example of an object: dog

**Inheritance:**

**I**nheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

**Encapsulation:**

Encapsulation is defined as the wrapping up of data under a single unit. Encapsulation can be achieved byDeclaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.

**Abstraction:**

Data Abstraction is the property by virtue of which only the essential details are displayed to the user. An abstract class can have parametrized constructors and default constructor is always present in an abstract class.

**Access Modifiers:**

Java’s access modifiers are **public**, **private**, and **protected**.

**Non-access modifiers :** In java, we have 7 non-access modifiers.

[**Overriding**](https://www.geeksforgeeks.org/overriding-in-java/) : Overriding is a feature of OOP languages like Java that is related to run-time polymorphism. A subclass (or derived class) provides a specific implementation of a method in the superclass (or base class).

[**Overloading**](https://www.geeksforgeeks.org/overloading-in-java/) : Overloading is also a feature of OOP languages like Java that is related to compile-time (or static) polymorphism.

**String:**

String is a sequence of characters. In java, objects of String are immutable which means a constant and cannot be changed once created.

**StringBuffer**is a peer class of **String**that provides much of the functionality of strings. String represents fixed-length, immutable character sequences while StringBuffer represents growable and writable character sequences.

The **StringBuilder** in Java represents a mutable sequence of characters.

The **Exception Handling in Java** is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.

### 1) Checked Exception

The classes which directly inherit Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

### 2) Unchecked Exception

The classes which inherit RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time, but they are checked at runtime.

### 3) Error

Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.Exception

**Java Exception Keywords:**

**Try :** The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.

**Catch :** The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.

**Finally :** The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.

**Throw :** The "throw" keyword is used to throw an exception.

**Throws :** The "throws" keyword is used to declare exceptions. It doesn't throw an exception. It specifies that there may occur an exception in the method. It is always used with method signature.