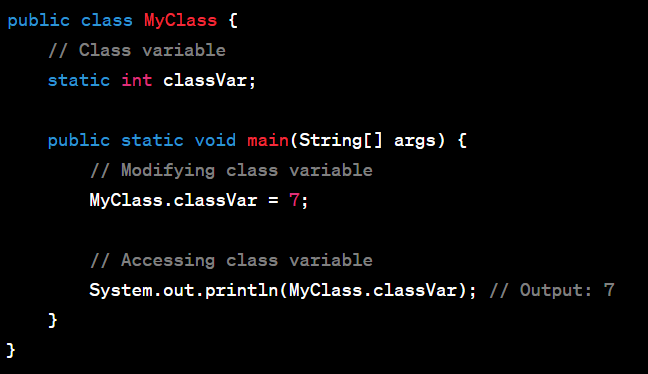
OOPS NOTES

**CLASS**

* A class is a blueprint from which individual objects are created.
* A class can contain any of the following variable types:-

1. **Local variables** − Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed.
2. **Instance variables** − Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class.
3. **Static variables** − Class variables are variables declared within a class, outside any method, with the static keyword.It can be accessed without making any object.



* A class can have the following types of access modifiers

**Public:**

1. Classes, methods, and fields marked as **public** are accessible from any other class.
2. There are no restrictions on accessing **public** members from outside the class or package.

**Default (Package-Private):**

1. If no access modifier is specified (default), the class, method, or field is accessible only within the same package.
2. It is more restrictive than **protected** and less restrictive than **private**

* A top-level class as **private** or **protected** will result in a compilation error.
* However, you can use access modifiers for inner or nested classes.
* A PRIVATE inner class can only be accessed in its own outer class.
* a class may include keywords like "**super**" if you use a superclass,
* "**implements**" if you are inheriting members, methods, and instances from the same class, and
* "**interface**" if you are inheriting any members, methods, and instances from a different class
* Final Class- a **final** class is a class that cannot be subclassed or extended by other classes. When a class is declared as **final**, it means that its design and implementation are considered complete, and it is not intended to be extended further.

**TYPES OF CLASS(based on nature and purpose)**

* 1. **Concrete Class-** A concrete class is a class that can be instantiated, meaning you can create objects of that class.

**Characteristics:**

* + Can have fields, methods, constructors, and other members.
  + Can be instantiated using the **new** keyword.
  + Can provide a complete implementation for all its methods.
  1. **Abstract Class-** An abstract class is a class that **cannot be instantiated** on its own and may contain abstract methods (methods without a body) that must be implemented by its subclasses.
  2. **Interfaces-** An interface is a collection of abstract methods. It is a way to achieve multiple inheritance in Java, as a class can implement multiple interfaces. Contains only abstract methods (prior to Java 8) or abstract methods and default methods (Java 8 and later).

**OBJECTS**

* an object is an instance of a class, created using the **new** keyword followed by a constructor.
* There are three stages involved in creating an object:-
  1. Declaration
  2. Instantiation
  3. Initializations

**CONSTRUCTOR**

* In Java, a Constructor is a block of codes similar to the method. It is called when an instance of the class is created.
* Each time an object is created using a **new()** keyword, at least one constructor (it could be the default constructor) is invoked to assign initial values to the **data members**of the same class.
* A constructor in Java can not be abstract, final, static, or Synchronized.

**TYPES OF CONSTRUCTOR**

1. **Default Constructor -** A constructor that has no parameters is known as default the constructor. A default constructor is invisible.

### ****Parameterized Constructor-**** A constructor that has parameters is known as parameterized constructor. If we want to initialize fields of the class with our own values, then use a parameterized constructor.

**Note-**  we can overload constructors for creating objects in different ways. The compiler differentiates constructors on the basis of the number of parameters, types of parameters, and order of the parameters.

### Copy Constructor- Unlike other constructors copy constructor is passed with another object which copies the data available from the passed object to the newly created object.

### Eg. -

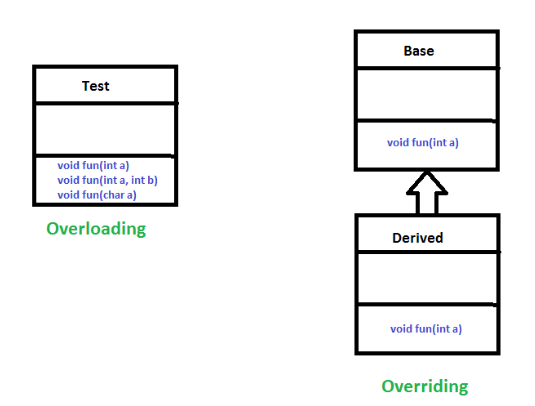
* **Con**structors can be only public and protected

**METHOD OVERRIDING**

* Method overriding is a concept in object-oriented programming (OOP) where a subclass provides a specific implementation for a method that is already defined in its superclass.
* **JISKA REFRENCE USSKA HI METHOD .**
* The access modifier of the overriding method in the subclass should be the same as or more permissive than the access modifier of the overridden method in the superclass
* Private, static members cannot be overridden
* Final method cannot be overridden

**METHOD OVERLOADING**

* When there are multiple functions with the same name but different parameters then these functions are said to be **overloaded**. Functions can be overloaded by changes in the number of arguments or/and a change in the type of arguments.

****

**INHERITENCE (IS A relationship)**

* Inheritance is a mechanism by which a new class (subclass or derived class) can acquire the properties and behaviors of an existing class (superclass or base class). The subclass can extend or override the functionality of the superclass.
* To inherit the parent class, a child class must include a keyword called "extends."
* Cyclic inheritence is invalid.
* Private members cannot be overriden
* Final class cannot be overriden

NEED FOR INHERITENCE

* The process of inheritance involves reusing the methods and data members defined in the parent class.

INHERITENCE AND ACCESS MODIFIERS

1. Default - The default access specifier is similar to the public access modifier. The difference is that the parent class is only accessible inside the Java project's package that is currently in use, but not outside of the package.
2. Public - The public access modifier is used when the user wants all child classes to have access to the parent class from anywhere in the Java project, as well as any other packages.
3. Protected - This class is accessible to its child class only. If a user needs to access the protected data members and methods from a different class, accessing the protected class’s child class is the only way it can be done.
4. Private - The private access specifier provides access within the class, but not outside of the class.

SUPER KEYWORD-

* the **super** keyword is used to refer to the immediate parent class of a subclass.
* Super keyword can be used to invoke the constructor and method of parent class.
* JAVA **DOESNOT** PROMOTES MULTIPLE INHERITENCE.

**POLYMORPHISM**

* The word polymorphism means having many forms.
* Two types of polymorphism:-

1. Compile-time Polymorphism
2. Runtime Polymorphism
3. RUN TIME POLYMORPHISM

* object is bound with their functionality at the run time
* Also known as method overriding or late binding
* Involves a base class and a derived class, where the derived class provides a specific implementation for a method declared in the base class.
* The correct method is determined during runtime based on the actual type of the object.

1. COMPILE TIME POLYMORPHISM

* It is also known as static polymorphism. This type of polymorphism is achieved by function overloading or operator overloading.
* here,object is bound with their functionality at the compile-time

**ABSTRACTION**

* **Data Abstraction** is the property by virtue of which only the essential details are displayed to the user.
* abstraction is achieved by [interfaces](https://www.geeksforgeeks.org/interfaces-in-java/) and [abstract classes](https://www.geeksforgeeks.org/abstract-classes-in-java/). We can achieve 100% abstraction using interfaces.
* abstract and static cannot be together
* public, default, and protected can be made abstract
* if class A is abstract and class B is also abstract then there is not need of implementation (overriding)

ABSTRACT CLASSES

* An abstract class is a class that is declared with an [abstract keyword.](https://www.geeksforgeeks.org/abstract-keyword-in-java/)
* An abstract class may or may not have all abstract methods. Some of them can be concrete methods
* Object cannot be created of abstract class.

ABSTRACT METHOD

* A method without body(no implementation) is known as abstract method.
* If a method is abstract then its class must be abstract.

INTERFACES

* In interface all the methods are abstract.(100% abstraction).
* Supports multiple inheritence
* Interfaces are public by default.
* “implements” keyword is used.
* Starting from Java 8, interfaces can have default and static methods with implementations.
* FUNCTIONAL INTERFACE- A functional interface is an interface that contains only one abstract method.
* Functional interfaces are used in conjunction with lambda expressions to provide concise ways to implement single-method interfaces.
* no object possible but can have reference variable
* variables are final and static in interface.
  + MARKER INTERFACE

COMPOSITION, AGGREGATION