**Class and Object**

**What are classes in Java?**

* Class is a template or a blueprint
* it is a logical entity.
* Class is a container.
* Class is a collection of object.
* Class does not occupy space.
* Class defines the variables and methods common to all objects of a certain kind.
* **When you create an object from a class, you are said to have created an instance[Object] of the class.**

**Example:**

* LAPTOP-monitor, keyboard, mousepad, charger.
* BANK-manager, clerk, cashier, ATM machine.

**Class Declaration**

|  |
| --- |
| class MyClass  {  // field, constructor, and  // method declarations  } |

**What are objects?**

* **Object is a real-time entity.**
* **Object is combination of states and behaviours.**
* Objects is an instance of class; each object stores about what it currently looks like.
* Object is a memory reference of a class.
* Object occupy space.

**What is Variable:**

**Variable are container for storing data values**

**Declaring Member Variables**

There are several kinds of variables:

* Member variables in a class—these are called fields.
* Variables in a method or block of code—these are called local variables.
* Variables in method declarations—these are called parameters.

**Methods in Java**

**What is method?**

* A **method** is a block of code or collection of statements or a set of code grouped together to perform a certain task or operation.
* It is used to achieve the **reusability** of code.
* We write a method once and use it many times. We do not require to write code again and again.
* The method is executed only when we call or invoke it.

## 

## **Types of Method**

There are two types of methods in Java:

* Predefined Method
* User-defined Method

**Method Overloading**

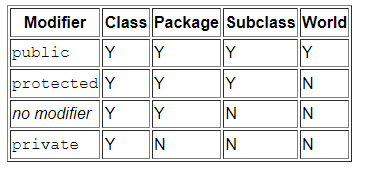
* If a class has multiple methods having same name but different in parameters, it is known as **Method Overloading**.
* Method overloading increases the readability of the program.
* The runtime polymorphism can be achieved by method overriding.
* Compile time Polymorphism is achieved by Method overloading.

**Access Modifier:**

In Java, access specifiers are the keywords which are used to define the access scope of the method, class, or a variable.

In Java, there are four access specifiers.

* **Public:** The classes, methods, or variables which are defined as public, can be accessed by any class or method.
* **Protected:** Protected can be accessed by the class of the same package, or by the sub-class of this class, or within the same class.
* **Default (Package-private):** Default are accessible within the package only.
* By default, all the classes, methods, and variables are of default scope.
* Default Access Modifier is package-private.
* **Private:** The private class, methods, or variables defined as private can be accessed within the class only



Access modifier are not at all related to local variable.

**private:** fields, method, class, constructor

**package-private**: fields, method, class, constructor

**public:** fields, method, class, constructor

**protected:** Outside package: child class present in other package can access protected variables and methods.

* Declaring the Method as private can be accessed within the class.
* Access Modifier is not allowed to local Variables but it can be final.
* Protected Method can be accessed by same package of same class and Other class or sub-class.
* Same package classes with Private methods and variable cannot be accessed by each other class.

**This Keyword:**

**Constructor**

**what is constructor?**

* constructor is useful for initializing object specific values.
* It is created using the same name of our class name.
* constructor area is getting called automatically when object/instance is created. [object-instance] [class is instantiated]
* constructor doesn't have any return type.
* Default constructor is invisible.
* This default constructor will call the no-argument constructor of the superclass.
* The compiler automatically provides a no-argument, default constructor for any class without constructors.
* Once we overload constructor, it becomes visible.
* Once we have visible constructor, there is no, no-argument constructor.
* Constructor can be private just as private method, variable, class etc.

Fields will be assigned with default values.

byte, short, int, long: 0

float, double: 0.0

boolean: false

char: ’’ (empty)

String: null

By default, Parameterized constructor in child class can call the no-argument constructor.

**This keyword:**

* it refer to the current object of same class.
* this can be used only inside non-static[object specific] area or context.
* inside static blocks, static methods we cant use this keyword.

**Private Methods:**

 We call the private method from outside the class by changing the runtime behaviour of that class.

(2 Warning will cause).

|  |
| --- |
| //import java.lang.reflect.Method;    class A {  private void display()  {  System.out.println("private method is invoked");  }  }    public class PrivateExample4{  public static void main(String[] args)throws Exception{    Class c = Class.forName("A");  Object o= c.newInstance();  Method m =c.getDeclaredMethod("display", null);  m.setAccessible(true);  m.invoke(o, null);  }  } |

|  |
| --- |
| PrivateExample4.java:15: warning: non-varargs call of varargs method with inexact argument type for last parameter;  Method m =c.getDeclaredMethod("display", null);  ^  cast to Class for a varargs call  cast to Class[] for a non-varargs call and to suppress this warning  PrivateExample4.java:17: warning: non-varargs call of varargs method with inexact argument type for last parameter;  m.invoke(o, null);  ^  cast to Object for a varargs call  cast to Object[] for a non-varargs call and to suppress this warning  Note: PrivateExample4.java uses or overrides a deprecated API.  Note: Recompile with -Xlint:deprecation for details.  Note: PrivateExample4.java uses unchecked or unsafe operations.  Note: Recompile with -Xlint:unchecked for details.  **2 warnings** |

|  |
| --- |
| **Output:**  C:\Users\ashik\Desktop\Java>java PrivateExample4  private method is invoked |

If the Constructor is Private we Cannot create Instance[Object] of the class, but the Constructor can be Private.

If the method is Private we can access Only within the class, not by sub-class of Same Package and Other Package

Yes,it is possible access the private methods of a class using **java reflection package**.

|  |
| --- |
| import java.lang.reflect.\*;  class Dummy{  private void foo(){  System.out.println("hello foo()");  }  }  class Test{  public static void main(String[] args) throws Exception {  Dummy d = new Dummy();  Method m = Dummy.class.getDeclaredMethod("foo");  //m.invoke(d);// throws java.lang.IllegalAccessException  m.setAccessible(true);// Abracadabra  m.invoke(d);// now its OK  }  } |

If the method is Protected we can access within the class and sub-class

**Naming Convention**

|  |  |  |
| --- | --- | --- |
| **Identifier** | **Naming Rules** | **Example** |
| **Class** | The first Letter should be Uppercase.  It should be a noun (a person, thing, place) such as Car, Home, Office, Town etc.  Use proper word instead of acronym. | public class **Parent**  **{**  **//code of snippet**  } |
| **Interface** | Same as class, First Letter should be Uppercase.  It should be an adjective (describes noun). | interface **Printable**  **{**  **//code of snippet**  } |
| **Method** | It should start with Lower case.  It should be a verb (meaning full word) such as work(), study() etc. | void **work()**  {  //code of snippet  } |
| **Variable** | it should start with a Lowercase letter such as salary, name etc.  It should not start with the special characters like & (ampersand), $ (dollar), \_ (underscore).  If the name contains multiple words, start it with the lowercase letter followed by an uppercase letter such as firstName, lastName. | // variable  int id; |
| **Constant** | It should be in Uppercase letters such as PI, MIN.  If the name contains multiple words, it should be separated by an underscore(\_) such as MAX\_PRIORITY.  It may contain digits but not start with digit. | //constant  static final int Min=25; |
| **Package** | It should be a Lowercase letter such as java, util,lang.  If the name contains multiple words, it should be separated by dots (.) such as java.util, java.lang. | tamilnadu.chennai |