1. Class vs Object

* **Class** is declared by class keyword. For ex. Class thread {};

**Object** is declared by new keyword. For ex. Thread t=new Thread ();

* **Class** do not allocate memory on time it is created while

**Object** allocates memory on time it is created.

* **Class** in java can be created in only one way that is by using class keyword while

**Object** in java can be created in multiple ways such as using new keyword, clone method, new Instance method.

1. Class variable vs Object variable

* Class variables also known as static variables are declared with the static keyword in a class but outside a method while object variable is declared in a class, but outside a method.
* Class variable is generally created when the program begins to execute while object variable is generally created when an instance of the class is created.
* Changes that are made to these variables through one object will reflect in another object while Changes that are made to these variables through one object will not reflect in another object.

1. Class method vs Object method

* An object is an instance of a class while a class is a group of objects which have common properties.
* An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc.

**Code:**

public class Main {  
 static void mymethod(){  
 System.*out*.println(" this is example of class method");  
 }  
 public static void main(String[] args) {  
 System.*out*.printf("Hello and welcome!");  
 }  
}

public class objectEx {  
public int area(int l,int b){  
 int result=l\*b;  
 return result;  
}  
}

public class objectMethodUSe {  
 public static void main(String[] args) {  
 int area=new objectEx().area(2,5);  
 System.*out*.println("are is "+ area);  
 }  
}

1. Keywords

* Keywords are reserved words that act as a key to a code. These are predefined words which cannot be used as a class name or variable.
* There are 52 keywords in java. Some main keywords are abstract, break, class, catch, continue, const, default, do, for, new, private, public, protected, return, switch, static, super, this, through etc.

1. Abstract

* Abstract is a reserve keyword. It is used to declare an abstract class.
* Abstract keyword can only be used as class and method.
* If we extend an abstract class in a non-abstract class then we must have to add method of abstract class in non-abstract class

**Code:**

public abstract class Threads {  
  
 abstract void run ();  
  
}  
  
class deiThread extends Threads{  
  
 @Override  
 void run() {  
 System.*out*.println("This is running");  
 }  
  
 public static void main(String[] args) {  
 deiThread dt=new deiThread();  
 dt.run();  
 }  
}

1. Implement

* In java, implement keyword is used to implement interface.
* To access method of interface we use implement keyword

**Code:**

public interface collage {  
 public void faculty();  
 public void branch();

}

public class student implements collage{  
 @Override  
 public void faculty() {  
 System.*out*.println("I am from Engineering faculty");  
 }  
  
 @Override  
 public void branch() {  
 System.*out*.println("I am from Electrical branch");  
  
 }  
  
 public static void main(String[] args) {  
 student st=new student();  
 st.faculty();  
 st.branch();  
 }  
}

1. Extends

* Extends keyword is used to inherit property of parent class to base class using inheritance.
* Parent class (super class)is that class which is being inherit and sub class(child class) is that class which inherit from another class.

**Code:**

public class extends\_use {  
 public static void main(String[] args) {  
 hello hl=new hello();  
 System.*out*.println("derived class "+hl.str1);  
 System.*out*.println("parent class "+hl.str);  
  
 }  
}  
class hi{  
 String str="hii";  
}  
  
class hello extends hi{  
 String str1="hey";  
}

1. Static

* **Static keyword** in java is used for memory management mainly.
* It is used to call class than object.
* No object is needed to call static.

**Code:**

public class static\_use {  
 int a=0;  
 static\_use(){  
 a++;  
 System.*out*.println(a);  
 }  
 public static void main(String[] args) {  
 static\_use su=new static\_use();  
 //static\_use su1=new static\_use();  
 }  
}

1. Public

* Public keyword is a reserve keyword in java.
* Public keyword is a type of access modifier which allow to get access everywhere inside and outside class and package.

**Code:**

class A{  
 public void message(){  
 System.*out*.println(" first message");  
 }  
}  
  
 class B extends A{  
 public void message(){  
 System.*out*.println(" first message overridden");  
 }  
}  
 public class public\_use {  
 public static void main(String[] args) {  
 B b=new B();  
 b.message();  
 }  
}

1. Protected

* The protected keyword is a type of access modifier.
* It is used for attributes, methods, and constructors, making them accessible in the same package and subclasses.

**Code:**

protected String message1="first protected message.";// in class A

public class protected\_use {  
 public static void main(String[] args) {  
 A a=new A();  
 System.*out*.println(a.message1);  
 }  
}

1. Private

* Private access modifier can be access in same class only.
* We can not assign private access modifier to outside class or package.

**Code:**

class A1{  
 private String str="This is private string";  
}  
public class private\_use {  
 //private String str1="This is private string";  
 public static void main(String[] args) {  
 A1 a=new A1();  
 System.*out*.println(a.str);  
   
 }  
}

1. Constructor

* In Java, a constructor is a block of codes like the method.
* It is a special type of method which is used to initialize the object.
* Constructor name must have same name as class name.
* A Java constructor cannot be abstract, static, final, and synchronized

**Code:**

public class constructor\_use {  
 constructor\_use(){  
 System.*out*.println("hi this is constructor.");  
 }  
 public static void main(String[] args) {  
 constructor\_use cs=new constructor\_use();  
  
 }  
}

1. Heap memory vs stack memory

* Stack memory stores items that have a very short life such as methods, variables, and reference variables of the objects while heap memory stores objects and Java Runtime Environment (JRE) classes.
* Stack memory follows LIFO order while heap memory do not follow any order.
* Stack memory is of smaller size and heap memory have larger size.
* Stack memory allow faster access while heap memory allow slower access.
* Stack memory is thread safe because each thread has its own stack while heap memory is not thread safe so properly synchronization of code is required.