**Static Members**

**Example 1: Modify static variable in main method and other methods of the class**

class Main

{

    // static variables a and b

    static int a = 10;

    static int b;

    static void printStatic()

    {

        a = a /2;

        b = a;

        System.out.println("printStatic::Value of a : "+a + " Value of b : "+b);

    }

    public static void main(String[] args)

    {

       printStatic();

       b = a\*5;

       a++;

    System.out.println("main::Value of a : "+a + " Value of b : "+b);

    }

}

**Example 2: Static variable as counter**

class Counter

{

    static int count=0;//will get memory only once and retain its value

    Counter()

    {

        count++;//incrementing the value of static variable

        System.out.println(count);

    }

}

class Main

{

     public static void main(String args[])

    {

        System.out.println("Values of static counter:");

        Counter c1=new Counter();

        Counter c2=new Counter();

        Counter c3=new Counter();

    }

}

**Example 3: Static Method**

class Main

{

    // static method

static void static\_method()

    {

        System.out.println("Static method in Java...called without any object");

    }

public static void main(String[] args)

    {

        static\_method();

    }

}

**Example: Static Method Overloading**

public class Main {

public static void static\_method() {

        System.out.println("static\_method called ");

    }

public static void static\_method(String msg) {

       System.out.println("static\_method(string) called with " + msg);

    }

     public static void main(String args[])

    {

        static\_method();

        static\_method("Hello, World!!");

    }

}

**Example: Overriding not possible with static methods**

classBase\_Class {

    // Static method in base class which will be hidden in substatic\_displayclass

    public static void static\_display() {

          System.out.println("Base\_Class::static\_display");

    }

}

classDerived\_Class extends Base\_Class {

   public static void static\_display() {

         System.out.println("Derived\_Class::static\_display");

    }

}

public class Main {

  public static void main(String args[ ])  {

        Base\_Class obj1 = new Base\_Class();

        Base\_Class obj2 = new Derived\_Class();

        Derived\_Class obj3 = new Derived\_Class();

        obj1.static\_display();

        obj2.static\_display();

        obj3.static\_display();

    }

}

**Example: a static block example**

|  |
| --- |
| class Main  {      static int sum = 0;      static int val1 = 5;      static int val2;        // static block     static {          sum = val1 + val2;          System.out.println("In static block, val1: " + val1  + " val2: "+ val2 + " sum:" + sum);          val2 = val1 \* 3;          sum = val1 + val2;      }         public static void main(String[] args)      {          System.out.println("In main function, val1: " + val1  + " val2: "+ val2 + " sum:" + sum);      }  }  **Example: Multiple Static Blocks in Java (in sequence execution, valies of first block are overwritten by second block**  class JavaExample2{  static int num;  static String mystr;  //First Static block  static{  System.out.println("Static Block 1");  num = 68;  mystr = "Block1";  }  //Second static block  static{  System.out.println("Static Block 2");  num = 98;  mystr = "Block2";  }  public static void main(String args[])  {  System.out.println("Value of num: "+num);  System.out.println("Value of mystr: "+mystr);  }  }  **Example: Static method accessed directly in static and non-static method** |

class JavaExample{

static int i = 100;

static String s = "Beginnersbook";

//Static method

static void display()

{

System.out.println("i:"+i);

System.out.println("i:"+s);

}

//non-static method

void funcn()

{

//Static method called in non-static method

display();

}

//static method

public static void main(String args[])

{

JavaExample obj = new JavaExample();

//You need to have object to call this non-static method

obj.funcn();

//Static method called in another static method

display();

}

}

class Human{

....

}

class Boy extends Human{

public static void main( String args[]) {

/\*This statement simply creates an object of class

\*Boy and assigns a reference of Boy to it\*/

Boy obj1 = new Boy();

/\* Since Boy extends Human class. The object creation

\* can be done in this way. Parent class reference

\* can have child class reference assigned to it

\*/

Human obj2 = new Boy();

}

}

**Example: Static Binding**

class Human{

public static void walk()

{

System.out.println("Human walks");

}

}

class Boy extends Human{

public static void walk(){

System.out.println("Boy walks");

}

public static void main( String args[]) {

/\* Reference is of Human type and object is

\* Boy type

\*/

Human obj = new Boy();

/\* Reference is of HUman type and object is

\* of Human type.

\*/

Human obj2 = new Human();

obj.walk();

obj2.walk();

}

}

**Example: Dynamic binding**

class Human{

//Overridden Method

public void walk()

{

System.out.println("Human walks");

}

}

class Demo extends Human{

//Overriding Method

public void walk(){

System.out.println("Boy walks");

}

public static void main( String args[]) {

/\* Reference is of Human type and object is

\* Boy type

\*/

Human obj = new Demo();

/\* Reference is of HUman type and object is

\* of Human type.

\*/

Human obj2 = new Human();

obj.walk();

obj2.walk();

}

}

**Example: Memory Efficient Static Variable**

**class** Student{

**int** rollno;//instance variable

   String name;

**static** String college ="ITS";//static variable

   //constructor

   Student(**int** r, String n){

   rollno = r;

   name = n;

   }

   //method to display the values

**void** display (){System.out.println(rollno+" "+name+" "+college);}

}

//Test class to show the values of objects

**public** **class** TestStaticVariable1{

**public** **static** **void** main(String args[]){

 Student s1 = **new** Student(111,"Karan");

 Student s2 = **new** Student(222,"Aryan");

 //we can change the college of all objects by the single line of code

 //Student.college="BBDIT";

 s1.display();

 s2.display();

 }

}

**Example: Counter without static variable**

//Java Program to demonstrate the use of an instance variable

//which get memory each time when we create an object of the class.

**class** Counter{

**int** count=0;//will get memory each time when the instance is created

Counter(){

count++;//incrementing value

System.out.println(count);

}

**public** **static** **void** main(String args[]){

//Creating objects

Counter c1=**new** Counter();

Counter c2=**new** Counter();

Counter c3=**new** Counter();

}

}

**Example: a static method that performs a normal calculation**

**class** Calculate{

**static** **int** cube(**int** x){

**return** x\*x\*x;

  }

**public** **static** **void** main(String args[]){

**int** result=Calculate.cube(5);

  System.out.println(result);

  }

}

**Example: Static Block Evocation**

**class** A2{

**static**{System.out.println("static block is invoked");}

**public** **static** **void** main(String args[]){

   System.out.println("Hello main");

  }

}

**Example: Static Method Example**

class StaticTest {

// non-static method

int multiply(int a, int b){

return a \* b;

}

// static method

static int add(int a, int b){

return a + b;

}

}

public class Main {

public static void main( String[] args ) {

// create an instance of the StaticTest class

StaticTest st = new StaticTest();

// call the nonstatic method

System.out.println(" 2 \* 2 = " + st.multiply(2,2));

// call the static method

System.out.println(" 2 + 3 = " + StaticTest.add(2,3));

}

}