# Experiment No : 1

**Name :** Samadhan Chandrakant Kadam **Roll No :** 20244346

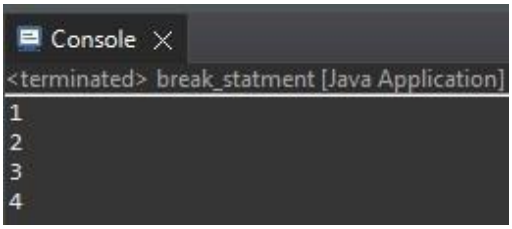
**Title :** Java Program Based on Branching And Looping Statements

## 1)Break statement:

**Program:**

```
package branching_statements;
public class break_statement {

	public static void main(String[] args )
	{int[]numbers ={1,2,3,4,5,6,7,8,9,10};
		for(int num : numbers)
				{
			if (num == 5)
			{
				break;
			}
			System.out.println(num);
				}
	}

}
```

## Output:




## 2)continue statement:

**Program:**

```
package branching_statements;
public class continue_statement
{
	public static void main(String[] args) {
		for (int i = 1; i<=10; i++)
		{
			if (i % 2 == 0)
			{
				continue;
```

```
                }
                System.out.println(i);
            }
        }

}
```

## Output:

```
Console
<terminated> Continue_statment [Java Application]
1
3
5
7
9
```

## 3)return statement:

**Program:**

```
package branching_statements;
public class return_statement
{
        public static void main(String[] args){
                int n = 16;
                long Factorial = calculatefactorial(n);
                System.out.println("factorial of "+n+" is "+Factorial);

        }
        public static long calculatefactorial(int n)
        {
                if (n == 0|| n == 1)
                {
                        return 1;
                }

        else
        {
                return n* calculatefactorial (n-1);
        }
    }
}
```

## Output:

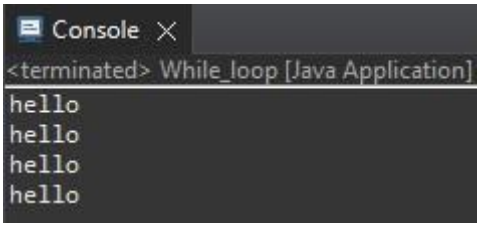
```
Console
<terminated> return_statment [Java Application]
factorial of 16is20922789888000
```

## 4)while loop:

**Program:**

```
package looping_statements;
public class while_loop
{
        public static void main(String[] args) {
                int i = 1;
        while (i<=5)
                {
                        System.out.println("Hello");
                        i++;
                }

        }

}
```

## Output:

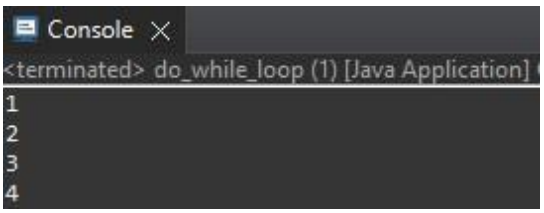



## 5)do-while loop:

**Program:**

```
package loop;
public classdo_while_loop
{ public static void main(String[] args) {
  int i=1; do
{
                        System.out.println
                        (i); i++;
                }while(i<=5);

        }

  }
```

## Output:

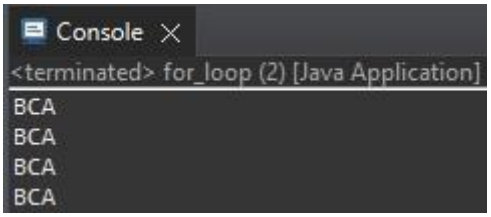


## 6)for loop:

**Program:**

```
package looping_statements;
public class for_loop {
	public static void main(String[] args) {
		int n = 5;
		for(int i = 1; i<=n; ++i)
		{
			System.out.println("BCA");
		}

	}

}
```

## Output:



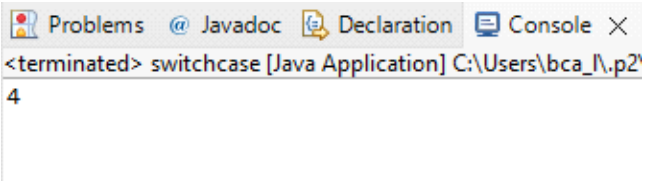

## 7)switch case:

**Program:**

```
package statement;
public class switch_case

{	public static void main(String[] args)

{

				int number = 4;
				switch(number)
		{



case 1: System.out.println("1");

break;


case 2: System.out.println("2");

break;


case 3: System.out.println("3");

break;
```

```
case 4: System.out.println("4");

break;


case 5: System.out.println("5");

break;

}
default : System.out.println("Number present in the 1 to5");



        }

}
```

**Output:**



# Experiment No : 2

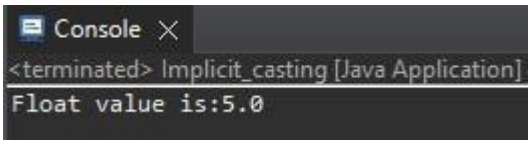
**Name:** Samadhan Chandrakant Kadam **Roll No:** 20244346

**Title :** Java program based on Type casting.

## Implicit Casting:

```
package Casting;

public class implicit_casting {

	public static void main(String[] args)
	{
		int a=5;
	float b=a;

		System.out.println("Float value is:"+b);

	}

}
```

## Output:




## Explicit Casting:

```
package Casting;
public class Emplicit_casting
{
	public static void main(String[] args)
	{
		double d=35.5;
		int  i=(int)d;
		float f=(float)d;
		long l=(long)d;
		short s=(short)d;
		byte b =(byte)d;

		System.out.println("Double value:"+d);
		System.out.println("int value:"+i);
		System.out.println("float value:"+f);
		System.out.println("long value:"+l);
		System.out.println("short value:"+s);
		System.out.println("byte value:"+b);
	}
```

}

**Output:**

```
Console
<terminated> Explicit_casting [Java Application]
Double value:35.5
int value.35
float value:35.5
long value:35
short value:35
byte value:+b
```

# Experiment No :3

**Name:** Samadhan Chandrakant Kadam **Roll No :** 20244346

**Title :** Java program based on command line arguments.

**Command line Argument:**

```
package cmd_line_arguments;
public class command_line_arguments
{ public static void main(String[] args)
 {
for(int i=0;
i<args.length;
i++)
 {
System.out.println("I Am "+args[i]);
 }
System.out.println("Total no of command line arguments is : "+args.length);



 }

 }
```

**Output :**

```
Problems  @ Javadoc  Declaration  Console ×
<terminated> command_line_arguments [Java Application] C
I Am samadhankadam
Total no of command line arguments is : 1
```

# Experiment No : 6

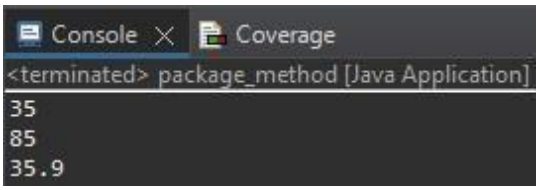
**Name:** Samadhan Chandrakant Kadam **Roll No:** 20244346

**Title :** Java program based on method overloading.

**Program:**

```
package method;
class SumDemo
{
	int sum(int x,int y)
	{
		return(x+y);
	}
	int sum(int x,int y,int z)
	{
		return(x+y+z);
	}
	double sum(double x,double y)
	{
		return(x+y);
	}

}

public class method_overloading {

	public static void main(String[] args)
	{
		SumDemo s=new SumDemo();
		System.out.println(s.sum(12,23));
		System.out.println(s.sum(15,20,50));
		System.out.println(s.sum(25.50,10.40));


	}

}
```

## Output:



# Experiment No : 7

**Name:** Samadhan Chandrakant Kadam **Roll No :** 20244346

**Title :** Java program based on method overriding.

**Program:**

```
package method;
class Super
{	int x;
Super(int x)
	{
		this.x=x;
	}
	void display()
	{
		System.out.println("Super x="+x);
	}
}
class Sub extends Super
{
	int y;
	Sub(int x,int y)
	{
		super(x);
this.y=y;
	}
	void display()
	{
		System.out.println("Super x="+x);
		System.out.println("Sub y="+y);
	}
}
public class method_overriding {

	public static void main(String[] args)
	{
		Sub s1=new Sub(200,300);
		s1.display();

	}

}
```

**Output:**

```
Console   Coverage
<terminated> method_overloading (1) [Java Application]
Super x=200
Sub y=300
```

# Experiment No : 8

**Name:**Samadhan Chandrakant Kadam **Roll No:**20244346

**Title :** Java program based on interfaces.

**Program**:

```
package inheritance;
class Student
{
	int roll_number;
	public void get_roll_number(int rn)
	{
		roll_number=rn;
	}
	public void put_roll_number()
	{
		System.out.println("Roll Number:-"+roll_number);
	}
}
class Test extends Student
{
	Double sem1_marks, sem2_marks;
	public void get_marks(double m1,double m2)
	{
		sem1_marks=m1;
sem2_marks=m2;
	}
	public void put_marks()
	{
		System.out.println("marks Obtained:");
		System.out.println("Semester-1:"+sem1_marks);
		System.out.println("Semester-2:"+sem2_marks);
	}
}
interface sports_marks
{
	double sports_points=2.0;
	public void put_sports_points();
}
class Result extends Test implements sports_marks
{
	double total_marks;
	public void put_sports_points()
```

```
        {
                System.out.println("Sports marks:"+sports_points);
        }
        public void Display()

{
                total_marks=sem1_marks+sem2_marks+sports_points;
                put_roll_number();
        put_marks();
                put_sports_points();
                System.out.println("Total marks:"+total_marks);
        }
}

public class multiple_inheritance {

        public static void main(String[] args)
        {
                Result ob=new Result();
ob.get_roll_number(123);
ob.get_marks(76.56,87.12);
                ob.Display();

        }
```

## Output:

```
Roll Number:-123
marks Obtained:
Semester-1:76.56
Semester-2:87.12
Sports marks:2.0
Total marks:165.68
```

# Experiment No : 9

**Name :** Samadhan Chandrakant Kadam **Roll No:** 20244346

**Title :** Java program based on packages.

**Program:**

```
package pack;

public class A

{
  public void msg()
{System.out.println("Hello");
}
}
```

**Package B:**

```
package Mypack;
import pack.*;
public class B
{
  public static void main(String args[])
  {
            A obj=new A();
            obj.msg();
  }
}
```

## Output:

# Experiment No : 11

**Name :** Samadhan Chandrakant Kadam **Roll No :** 20244346

**Title :** Java program based on Exception Handling.

## 1] try-catch block:

**Program:**

```
package java_exception;
 //try-catch block
public class JavaExceptionExample
{
        public static void main(String args[])
        {
try
                {
                        int data=100/0;
                }
                catch(ArithmeticException e)
                {
                        System.out.println(e);
                }

                        System.out.println("rest of the code....");

        }

}
```

```
java.lang.ArithmeticException: / by zero
rest of the code....
```

## Output:

## 2] try-catch-finally block:

**Program:**

```
package java_exception;

public class FinallyBlock
{
  public static void main(String args[])
  {
```

```
try
{
int data=25/5;
System.out.println(data);
}
catch(NullPointerException e)
{
System.out.println(e);
}
finally
{
System.out.println("finally block is always executed");
}
System.out.println("rest of the code... ");
  }
}
```

**Output:**

```
5
finally block is always executed
rest of the code....
```

# Experiment No : 4

**Name :** Samadhan Chandrakant Kadam **Roll No :** 20244346

**Title :** Java program based on constructors.

## Default Constructor:

**Program:**

```
Packageconstructo;
class
   Bike1
   { Bike1()
            {
            System.out.println("Bike is Created");
            }
   } public class
   default_constructor {

            public static void main(String[] args)
                     { Bike1 b=new Bike1();

            }

   }
```

**Output:**

```
Bike is Created
```

## Parameterized Constructor :

**Program:**

```
package constructor;
class Student4 {
  int id;
  String name; Student4(int i,String n)
 {
  id = i; name = n;
 }    void
display()
 {
  System.out.println(id+""+name);
```

```
 }

}
public class parameterized_constructor
{
        public static void main(String[] args) {
                Student4 s1 = new Student4(111 , " Karan ");
Student4 s2 = new Student4(222 , " Aryan ");
s1.display();

                s2.display();
                }

}
```

**Output:**

```
111 Karan
222 Aryan
```

# Experiment No : 5

**Name :**Samadhan Chandrakant Kadam **Roll No:**20244346

**Title :** Java program based on inheritance.

## Single inheritance:

**Program:**

```
package inheritance;
class Animal
{
	void eat()
	{
		System.out.println("eating...");
	}
}
class Dog extends Animal
{
	void bark()
	{
		System.out.println("barking...");
	}
}

public class single_inherritance
{
	public static void main(String[] args)
		{
			Dog d=new Dog();
	d.bark();	d.eat();
		}

	}
```

## Output:

```
barking...
eating...
```

## Multilevel inheritance:

**Program:**

```
package inheritance;
class Animal
{
	void eat()
	{
		System.out.println("eating...");
	}
}
class Dog extends Animal
{
	void bark()
	{
		System.out.println("barking...");
	}
}
class BabyDog extends Dog
{
	void weep()
	{
		System.out.println("weeeping");
	}
}
public class multilevel_inheritance {

	public static void main(String[] args)
	{
		BabyDog d=new BabyDog();
d.weep();
		d.bark();
		d.eat();
	}

}
```

## Output:

```
weeeping
barking...
eating...
```

## Hierarchical inheritance:

**Program:**

```
package inheritance;
class A
{
	public void print_A()
	{
		System.out.println("Class A");
	}
}
class B extends A
{
	public void print_B()
	{
		System.out.println("Class B inherits from class A");

	}
}
class C extends A
{
	public void print_C()
	{
		System.out.println("Class C inherits from class A");
	}
}
class D extends A
{
	public void print_D()
	{
		 System.out.println("Class D inherit from class A");
	}
} public class hierarchical_inheritance
{
	public static void main(String[] args)
	{
B obj_B=new B();		obj_B.print_A();
	obj_B.print_B();

C obj_C=new C();
		obj_C.print_C();

D obj_D=new D();
		obj_D.print_D();

	}
```

**Output:**

```
Class A
Class B inherits from class A
Class C inherits from class A
Class D inherit from class A
```

# Experiment No : 10

**Name :** Samadhan Chandrakant Kadam **Roll No :** 20244346

**Title :** Java program Based on Multithreading

**Program :**

```
package multithreading;
	class NumberPrinter extends Thread
	{
		private String threadName;
		public NumberPrinter(String name)
		{
			this.threadName = name;

		}
		public void run()

		{
			for(int i = 1; i<=5; i++)

			{
				System.out.println(threadName +"-Number:"+i);
				try
				{
					Thread.sleep(500);

				}catch(InterruptedException e)

				{
					System.out.println(threadName +"interrupted");
				}
			}

			System.out.println(threadName + "has finished executing");
		}
	}
	public class multithreading_program {
	public static void main(String[] args)
	{
		NumberPrinter thread1 = new NumberPrinter("Thread 1");
		NumberPrinter thread2 = new NumberPrinter("Thread 2");

		thread1.start();
		thread2.start();

		try {

			thread1.join();
```

```
                thread2.join();

            }catch(InterruptedException e) {

                System.out.println("All threads have finished executing");

            }
        }
}
```

## Output:

```
<terminated> multithreading_program (1) [Java Application] C:\Users\bca_I\.p2\pool\plugins\org.e
Thread 1-Number:1
Thread 2-Number:1
Thread 1-Number:2
Thread 2-Number:2
Thread 2-Number:3
Thread 1-Number:3
Thread 1-Number:4
Thread 2-Number:4
Thread 1-Number:5
Thread 2-Number:5
Thread 1has finished executing
Thread 2has finished executing
```

## Thread Life Cycle

**Program :**

```
package multithreading;
class MyThread extends Thread

 {
  public void run()
 {
        try {   System.out.println(Thread.currentThread().getName()+"is in Runnable
                Thread.sleep(100);
state");        synchronized (this)
                {
                        System.out.println(Thread.currentThread().getName() + " is in
Blocked state");        Thread.sleep(200);

                }
                System.out.println(Thread.currentThread().getName() + " is terminating");

        }
```

```
        catch(InterruptedException e)
        {
                System.out.println(Thread.currentThread().getName()+"was intrrepted");
        }
 }
 }
public class thread_life_cycle
{
        public static void main(String[]args)
        {
                MyThread thread1=new MyThread();
                MyThread thread2=new MyThread();

                System.out.println(thread1.getName()+"is in New state");
                thread1.start();

                System.out.println(thread2.getName()+"is in New state");
                thread2.start();

                try {
                        Thread.sleep(50);;
                        System.out.println("Main thread is in Waiting state");
                        synchronized(thread1)
                        {
                                thread2.join();

                        System.out.println(thread2.getName()+"has finished executing");
                }
                catch (InterruptedException e)
                {
                        System.out.println("Main thread was intrrupted");
                }
                System.out.println("Main thread was terminating");

        }
}
```

**Output:**

```
<terminated> thread_life_cycle [Java Application] C:\Users\bca_I\.p2\pool\p
Thread-0is in New state
Thread-1is in New state
Thread-0is in Runnable state
Thread-1is in Runnable state
Main thread is in Waiting state
Thread-1 is in Blocked state
Thread-1 is terminating
Thread-1has finished executing
Main thread was terminating
Thread-0 is in Blocked state
Thread-0 is terminating
```