## SESSION 6 PROGRAMS(OOPS Concepts)

### Program 1 Overloading of methods

```
//Class to demonstrate Overloading of methods or Compile Time Polymorphism
public class Mathematics {

	int result ;

	public int sum( int i , int j ) //same method name sum() with 2 parameters
	{
		result = i+j ; //output is different
		return result ;
	}

	public int sum( int i , int j , int k ) //same method name sum() with 3
parameters
	{
		result = i+j+k ; //output is different
		return result ;
	}

	public int sum(int i , int j ,int k , int l) //same method name sum() 4
parameters
	{
		result = i+j+k+l ; //output is different
		return result ;
	}


	public static void main(String[] args) {

		Mathematics mathsObj = new Mathematics();

		int output1= mathsObj.sum(10, 20);

		int output2= mathsObj.sum(10, 20 , 30);

		int output3= mathsObj.sum(10, 20 , 30 ,40);


		System.out.println("1st Sum is " + output1 );
		System.out.println("2nd Sum is " + output2 );
		System.out.println("3rd Sum is " + output3 );
	}

}
```

## Program2 Method Overriding

```
//Parent Class to show Inheritance & Method Overriding
public class BankAcct {


	//This Method will be overeridden in the child class
    void displayBankDetails()
	{
		System.out.println("Welcome to General Bank");
	}



}

//Child Class to show Inheritance & Method Overriding
public class SBIAcct extends BankAcct  {


	@Override  //Overridden Method
    void displayBankDetails()
	{
		System.out.println("Welcome to State Bank of India");//Changed code
different from PArent class

	}
```

## Program3 Abstract Class

```
//Class to demonstrate Use of Abstract Class
public abstract class Car {

	//Abstract Class can have constructor
	Car()
	{
		System.out.println("Constructor of Abstract Class is called");
	}

	//abstract method has no code
	abstract void accelerate();

	public static void main(String[] args) {

		//Car honda = new Car() ;
	}

}

public class Honda extends Car {

	Honda()
	{
		System.out.println("Constructor of Honda is called ");
	}

	public static void main(String[] args) {
		Honda hondaCity = new Honda();

	}

	@Override
	void accelerate() {
		System.out.println("Accelarating Honda");//Implementation of
abstract method

	}

}
```

}

## Program4 Interface

```
//Interface Example
public interface GearBox {

	String gearType="Manual"; // only constants allowed  no variables allowed
in interface

	void shiftGear();//only abstract methods allowed

}


public class Honda extends Car implements GearBox{

	Honda()
	{
		System.out.println("Constructor of Honda is called ");
	}

	public static void main(String[] args) {
		Honda hondaCity = new Honda();
		//hondaCity.gearType="Auto";

	}

	@Override
	void accelerate() {//this method is coming from Car Class
		System.out.println("Accelarating Honda");//Implementation of
abstract method

	}

	@Override
	public void shiftGear()
	{//this method is coming from GearBox interface
		System.out.println("Gear Shift is Manual");
	}
}
```

## SESSION 6 ASSIGNMENTS

1. **Write a Program to create a class Temperature**
   **Create method convertTemperature(int degrees)**
   **Print temperature in degrees**
   **. Create method convertTemperature(int degrees , int fahrenite)**
   **Print temperature in degrees & fahrenite**

2. **Create a parent class Company**
   **Create a method displayCompanySize()**
   **Print Company Size as 0**

   **Now Create a child class TCS**
   **Override the method displayCompanySize()**
   **Print Company Size as 1000**

   **Create main method**
   **Create object of TCS class in the main method.**
   **Call displayCompanySize() thru TCS Object**

   **Create object of Company class in the main method.**
   **Call displayCompanySize() thru Company Object**

3. **Create an Abstract Class College**
   **Create an abstract method displayCollegeName()**

   **Create a child class REC inheriting from Class College**
   **Override the abstract method displayCollegeName()**

   **Create main method**
   **Create object of REC class in the main method.**
   **Print College Name as "Regional Engineering College"**

4. **Create an interface PaymentGateway**
   **Declare int amount = 1000**
   **Create method void transferPayment()**

   **Create a Child class ServiceProvider implementing above interface**
   **Implement transferPayment()**