EXPERIMENT 11

// The phenomenon of Method Overriding

PROGRAM 13

Source Code -

```
class Parent
{
   private int ParentNo;
   public int num1;

   Parent()
   {     ParentNo=1; num1=10;      }

   int Add()
   {     return (ParentNo+num1);   }
}
class Child extends Parent
{
   private int ChildNo;

   Child()
   {     ChildNo=2;    }
   int Add()        //overriding Add() present in Parent class
   {     return (ChildNo+num1);      }
}
class MainClass
{
   public static void main(String args[ ])
   {
      Child c = new Child();

      System.out.println("Sum of parent class and child class numbers = " + c.Add());
   }
}
```

Output -

```
Sum of parent class and child class numbers = 12
```

PROGRAM 14

// Dynamic Method Dispatch - an alternative to method overriding

Source Code -

```
class Parent
{
   private int ParentNo;
   public int num1;

   Parent()
   {     ParentNo=1; num1=10;     }

   int Add()
   {     return (ParentNo+num1);   }
}
class Child extends Parent
{
   private int ChildNo;

   Child()
   {     ChildNo=2;    }
   int Add()       //overriding Add() present in Parent class
   {     return (ChildNo+num1);     }
}
class MainClass
{        public static void main(String args[ ])
         {        Parent p,ref;
                  p = new Parent();
                  Child c = new Child();

                  //Dynamic method dispatch
                  ref=p;
                  System.out.println(ref.Add());
                  ref=c;
                  System.out.println(ref.Add());       }        }
```

Output -

```
11
12
```

PROGRAM 15

// Abstract Class & Abstract Methods
- an alternative to method overriding

Source Code -

```
abstract class Parent
{
   private int ParentNo;
   public int num1;

   Parent()
   {   ParentNo=1; num1=10;    }

   abstract int Add();
}
class Child extends Parent
{
   private int ChildNo;

   Child()
   {   ChildNo=2;  }

   int Add()
   {   return (ChildNo+num1);  }
}
class MainClass
{
   public static void main(String args[ ])
   {
      Child c = new Child();

      System.out.println("Sum = " + c.Add());
   }
}
```

Output -

```
Sum = 12
```

VIVA QUESTIONS

1) **What is method overriding?**
   In a class hierarchy, when a method in a subclass has the same prototype as a method in its superclass but different definition, then the method in the subclass is said to override the method in the superclass. This is because when such a method is called from within the subclass, it will always refer to the version of that method defined by the subclass. The version of the method defined by the superclass will be hidden.
2) **What is the difference between method overloading and method overriding?**

Method Overloading	Method Overriding
Method overloading is a compile-time polymorphism.	Method overriding is a run-time polymorphism.
Method overloading helps to increase the readability of the program.	Method overriding is used to grant a specific implementation to a method, which was already defined in its parent class.
In method overloading, methods must have the same name but different type or number of arguments.	In method overriding, methods must have the same prototype.
Static binding is used for overloaded methods.	Dynamic binding is used for overriding methods

3) **What is an abstract class?**
   A super class that only defines a generalized form of data members and member functions, that will be shared by all of its subclasses, leaving it to each subclass to give proper implementation for them is known as an abstract class.