List of Concepts Involved:

- What is polymorphism?
- How to achieve polymorphism
- Runtime vs Compile time polymorphism
- Abstract keyword and Abstraction
- Abstract class and Abstract method
- final class
- final variable
- final method

What is polymorphism?

If one thing exists in more than one form then it is called Polymorphism.
Polymorphism is a Greek word, where Poly means many and morphism means structures or forms.

1.Static Polymorphism
2.Dynamic Polymorphism

1.Static Polymorphism:
If polymorphism exists at compilation time then it is called Static Polymorphism.
Ex: Overloading.

2.Dynamic Polymorphism:
If the polymorphism exists at runtime then that polymorphism is called Dynamic Polymorphism.
Ex: Overriding

Method Overriding:

The process of replacing existing method functionality with some new functionality is called Method Overriding.

To perform Method Overriding, we must have inheritance relationship classes.
In Java applications, we will override super class method with sub class method.
In Java applications,we will override super class methods with subclass methods.
If we want to override super class method with subclass method then both super class method and sub class method must have the same method prototype.

Steps to perform Method Overriding:

1. Declare a superclass with a method which we want to override.
2. Declare a subclass and provide the same super class method with different implementation.
3. In the main class,in the main() method,prepare an object for the subclass and prepare a reference variable for super class[UpCasting].
4. Access the super class method then we will get output from the sub class method.

Example

```
class Loan{
	public float getIR(){
		return 7.0f;
	}
}
class GoldLoan extends Loan{
	public float getIR(){
		return 10.5f;
	}
}
class StudyLoan extends Loan{
	public float getIR(){
		return 12.0f;
	}
}
class CraftLoan extends Loan{
}
class Test{
	public static void main(String[] args){
		Loan gold_Loan=new GoldLoan();
		System.out.println("Gold Loan IR :"+gold_Loan.getIR()+"%");

		Loan study_Loan=new StudyLoan();
		System.out.println("Study Loan IR :"+study_Loan.getIR()+"%");

		Loan craft_Loan=new CraftLoan();
		System.out.println("Craft Loan IR :"+craft_Loan.getIR()+"%");
	}
}
```

NOTE:

To prove method overriding in Java, we have to access the super class method but JVM will execute the respective sub class method and JVM has to provide output from the respective sub classmethod, not from superclass method. To achieve the above requirement we must create reference variables for only super classes and we must create objects for subclasses.

```
class A{
	 void m1(){
		 System.out.println("m1-A");
	 }
}
class B extends A{
	void m1(){
		 System.out.println("m1-B");
	}
}
public class Test{
	 public static void main(String args[]){
		A a  = new B();
		  a.m1();
	 }
}
```

Rules to perform Method Overriding:

1.To override super class method with sub class, then super class method must not be declared as private.

Ex:

```
class A{
        private void m1(){
                System.out.println("m1-A");
        }
}
class B extends A{
      void m1(){
              System.out.println("m1-B");
      }
}

public class Test{
        public static void main(String args[]){
                    A a=new A();
                    a.m1();
        }
}
```

2.To override super class method with sub class method then sub class method should have the same return type of the super class method.

EX:

```
class A{
      int m1(){
              System.out.println("m1-A");
              return 10;
      }
}
class B extends A{
      void m1(){
              System.out.println("m1-B");
      }
}
public class Test{
       public static void main(String args[]){
               A a=new B();
               a.m1();
       }
}
```

3.To override super class method with sub class method then super class method must not be declared as final sub class method may or may not be final.

EX:

```
class A{
      void m1(){
              System.out.println("m1-A");
      }
}
class B extends A{
        final void m1(){
               System.out.println("m1-B");
        }
}
public class Test{
      public static void main(String[] args){
             A a=new B();
            a.m1();
      }
}
```

4.To override superclass method with sub class method either super class method or subclass method as static then compiler will rise an error.If we declare both super and sub class method as static in method overriding compiler will not rise any error,JVM will provide output from the super class Method.

NOTE: If we are trying to override superclass static method with subclass static method then super class static method will override subclass static method,where JVM will generate output from super class static method.

EX:

```
class A{
       static void m1(){
              System.out.println("m1-A");
       }
}
class B extends A{
       static void m1(){
              System.out.println("m1-B");
       }
}
public class Test{
       public static void main(String args[]){
                     A a=new B();
                     a.m1();
       }
}
```

Inherited method

- The method which would come from parent to child due to inheritance is called inherited method.

Example

```
class Parent{
        public void methodOne(){System.out.println("methodOne from parent");}
}
class Child extends Parent{
          public void methodTwo(){System.out.println("methodTwo from child");}
}

public class TestApp{
       public static void main(String... args){
                Parent p=new Parent();
               p.methodOne();

                Child c=new Child();
               c.methodOne();//inherited method
                c.methodTwo();//Specialized method

                Parent p1=new Child();
                p1.methodOne();
                p1.methodTwo();//CE: can't find the symbol methodTwo in Parent
       }
    }
```

Overridden Method

The method which is taken from Parent and changes the implementation as per the needs of the requirement in the class is called the “overridden method”.

Example

```
class Parent{
    public void methodOne(){System.out.println("methodOne from parent");}
}
class Child extends Parent{
        @Override
        public void methodOne(){System.out.println("methodOne from child");}
 }
 public class TestApp{
        public static void main(String... args){
              Parent p=new Parent();
               p.methodOne();//methodOne from parent

             Child c=new Child();
              c.methodOne();//methodOne from child
        }
   }
```

Inherited method

- The method which would come from parent to child due to inheritance is called inherited method.

Example

```
class Parent{
        public void methodOne(){System.out.println("methodOne from parent");}
}
class Child extends Parent{
          public void methodTwo(){System.out.println("methodTwo from child");}
}

public class TestApp{
       public static void main(String... args){
                Parent p=new Parent();
               p.methodOne();

                Child c=new Child();
               c.methodOne();//inherited method
                c.methodTwo();//Specialized method

                Parent p1=new Child();
                p1.methodOne();
                p1.methodTwo();//CE: can't find the symbol methodTwo in Parent
       }
    }
```

Overridden Method

The method which is taken from Parent and changes the implementation as per the needs of the requirement in the class is called the “overridden method”.

Example

```
class Parent{
    public void methodOne(){System.out.println("methodOne from parent");}
}
class Child extends Parent{
        @Override
        public void methodOne(){System.out.println("methodOne from child");}
 }
 public class TestApp{
        public static void main(String... args){
              Parent p=new Parent();
               p.methodOne();//methodOne from parent

             Child c=new Child();
              c.methodOne();//methodOne from child
        }
   }
```

5.To override super class method with subclass method,sub class method must have either the same scope of the super class method or more scope when compared with super class method scope otherwise the compiler will raise an error.

EX:

```
class A{
      protected void m1(){
         System.out.println("m1-A");
      }
}
class B extends A{
      public void m1(){
             System.out.println("m1-B");
      }
}
public class Test{
       public static void main(String args[]){
              A a=new A();
              a.m1();
       }
}
```

6.To override super class method with subclass method subclass method should have either same access privileges or weaker access privileges when compared with super class method access privileges.

CompileTime Polymorphism vs RunTime Polymorphism

What are the differences between method overloading and method overriding?

1. The process of extending the existing method functionality with new functionality is called Method Overloading.
   The process of replacing existing method functionality with new functionality is called Method Overriding.

2. In the case of method overloading, different method signatures must be provided to the methods
   In the case of method overriding, the same method prototypes must be provided to the methods.

3. With or without inheritance we can perform method overloading
   With inheritance only we can perform Method overriding