Polymorphism:

Diff types of JVM memories:
1. Heap area--> non-static method declaration
2. Static pool area--> static method declaration
3. method area --> static & non-static method definition
4. stack --> main()--> method execution flow

Polymorphism:
It is one of the OOPs principles where one object showing different behaviour at different stages of life cycle.
Polymorphism is a Latin word where poly stand for many & morphism stands for forms.
In java Polymorphism is classified into 2 types:
1. Compile time Polymorphism
2. Runtime Polymorphism

1. Compile time Polymorphism:
In Compile time Polymorphism method declaration is going to get binded to its definition at compilation time, based on argument/input/parameter is known as compile time Polymorphism.

As binding takes during compilation time only, so it is also known as early binding.

//once binding is done, again rebinding can't be done, so it is called static binding.

Method overloading is an example of compile time Polymorphism.

2. Runtime Polymorphism:
In Runtime Polymorphism method declaration is going to get binded to its definition at Runtime/execution time, based on object creation is known as runtime Polymorphism.

As binding takes during Runtime/execution time, so it is also known as late binding.

//once binding is done, again rebinding can be done, so it is called dynamic binding.

Method overriding is an example of Runtime Polymorphism.

Method overloading:
Declaring multiple method with same method name but with different argument/parameter/inputs in a same class is called method overloading

Method overriding:
Acquiring super class method into sub class with the help of extends keyword & changing implementation/definition according to subclass specification is called method overriding

1: What is polymorphism?
2: Types of polymorphism?
3: What is compile time polymorphism?
4: What is runtime polymorphism?

5: can we overload static method?
    -> yes
5: can we overload main method? if yes then how?
    -> yes , create 2 main methods 1 with String [], 1 with int num
6: can we override static method in java?
    -> No
7: what is method hiding?
8: can we override main method?
    ->No
9: what is method overloading?
10: what is method overriding?
11: can we overload constructor?
    ->yes
12: what is constructor overloading?
13: can we override constructor?
    -> No
14: what is early binding?
15: what is late binding?
16: what is static binding?
17: what is dynamic binding?

```
package Polymorphism;
public class Sample1
{
        //method Overloading

        public void add(int a, int b)
        {
                System.out.println(a+b);
        }

        public void add(int a, int b, int c)
        {
                System.out.println(a+b+c);
        }
}
```

```
package Polymorphism;
public class TestOverloading
{
        public static void main(String[] args)
        {
                Sample1 s1=new Sample1();
                s1.add(10, 20);
                s1.add(5, 6, 7);
        }
}
```

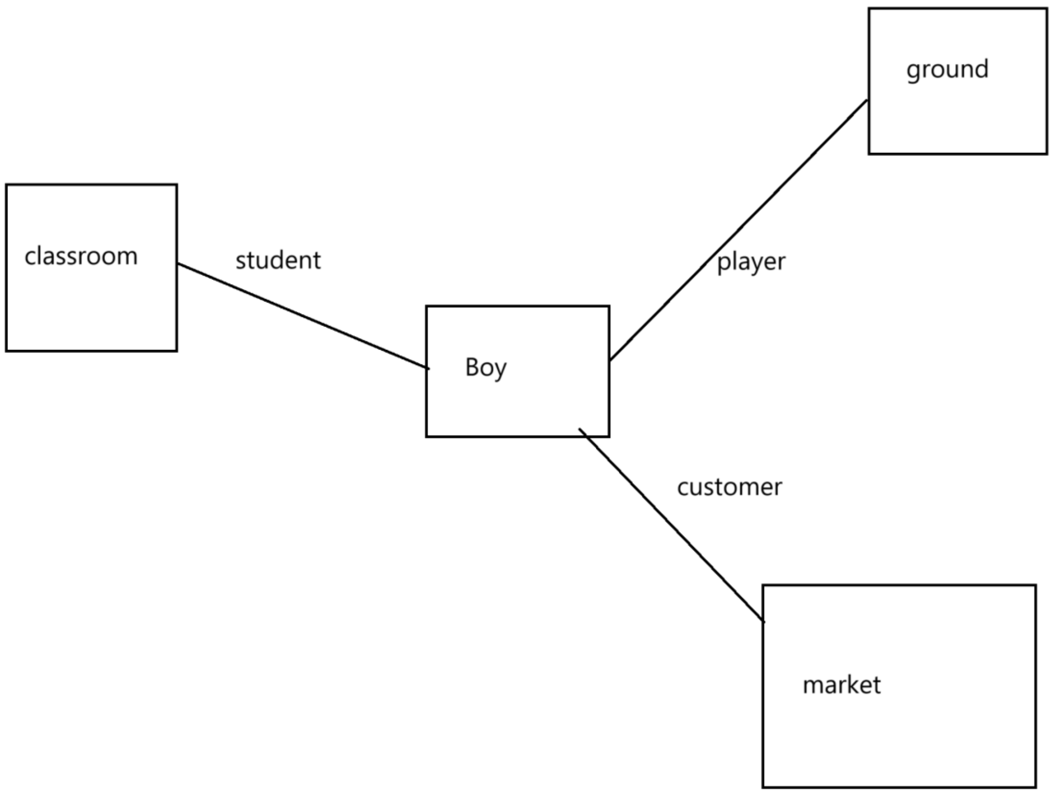
```
package Polymorphism;
//super class
public class Father
{
        public void car()
        {
                System.out.println("car: Kia Seltos");
        }
```

```
		public void money()
		{
				System.out.println("money: 1L");
		}

		public void home()
		{
				System.out.println("home: 2BHK");
		}
}

package Polymorphism;
//sub/ child class
public class Son extends Father
{
		//method overriding

		public void mobile()
		{
				System.out.println("Mobile: Samsung S20");
		}

		public void car()	//method overriding
		{
				System.out.println("car: BMW");
		}

		public void money()	//method overriding
		{
				System.out.println("money: 10k");
		}

//		public void home()
//		{
//				System.out.println("home: 2BHK");
//		}

}

package Polymorphism;
public class TestOverriding
{
		public static void main(String[] args)
		{
				Son s=new Son();
				s.mobile();
				s.car();
				s.money();
				s.home();
		}
}
```

```
package Polymorphism;
public class demo1
{
		//example of main method overloading

		public static void main(String[] args)
		{

				main(10);
		}


		public static void main(int num1)
		{
				System.out.println(num1);
				System.out.println("running int main method");
		}

}
```

4: stack //decides method execution flow main()	1: Heap Area non-static method declaration

3: Method Area
 	
// non-static method definition
 	
//static method definition
	2: static pool area
	
	static method declaration


classroom
student
Boy
player
ground
customer
market