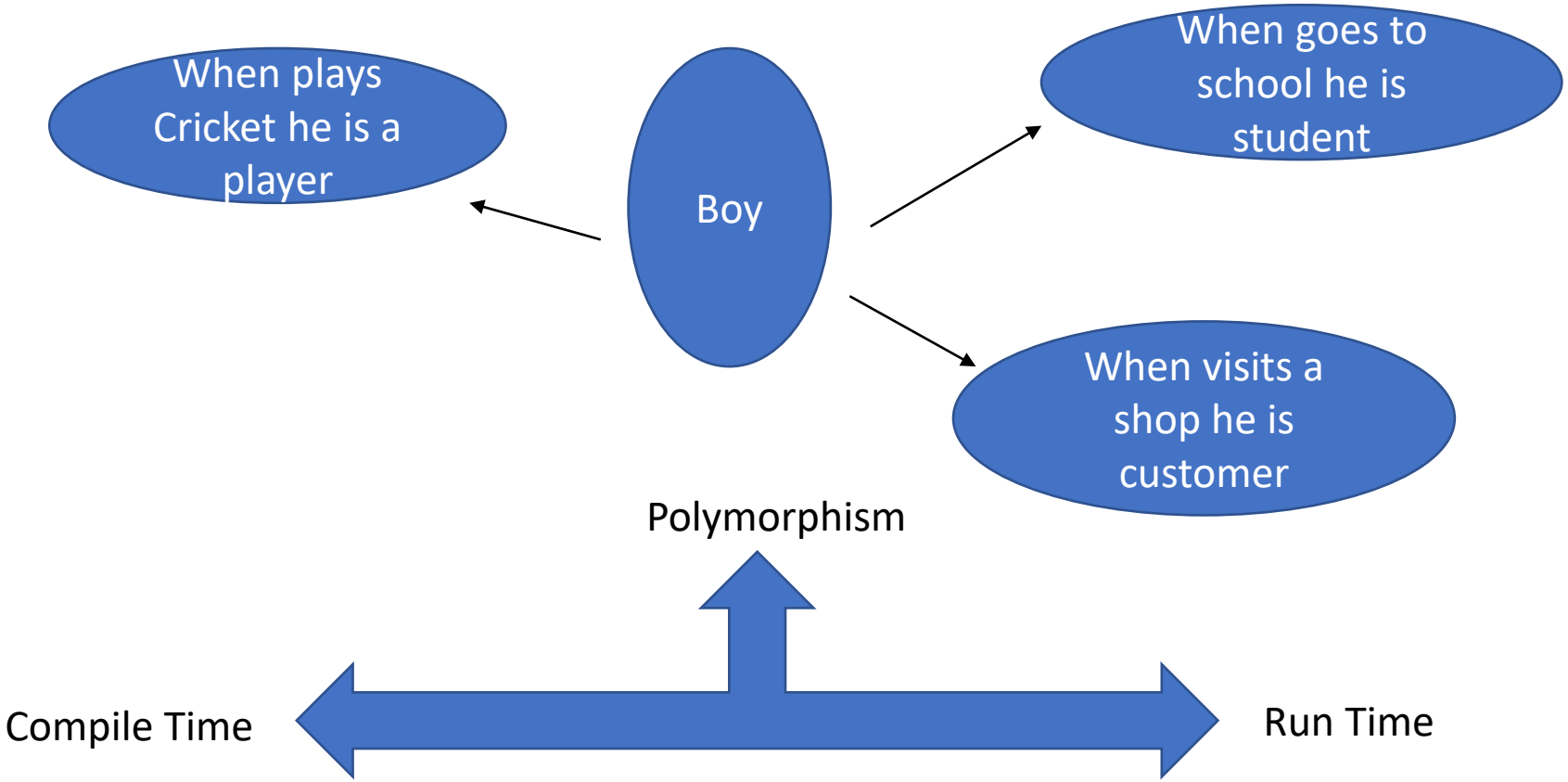
Java Basics

Polymorphism:-

Polymorphism in Java is a concept by which we can perform a *single action in different ways*. Polymorphism is derived from 2 Greek words: poly and morphs. The word "poly" means many and "morphs" means forms. So polymorphism means many forms.

There are two types of polymorphism in Java: compile-time polymorphism and runtime polymorphism. We can perform polymorphism in java by method overloading and method overriding.



When a single element perform different actions in different stages of lifecycle is known as polymorphism

- Compile Time Polymorphism :-

In compile time polymorphism method declaration gets binded to its body (definition) at the time of compilation, based on parameters and arguments is called as compile time polymorphism. Method overloading is the example of compile time polymorphism.

Ex:-

```
Public class Demo{
        void test(){
        zero argument
                }

        void test(int a){
        int argument
                }
```

Compile time polymorphism is called as early binding.

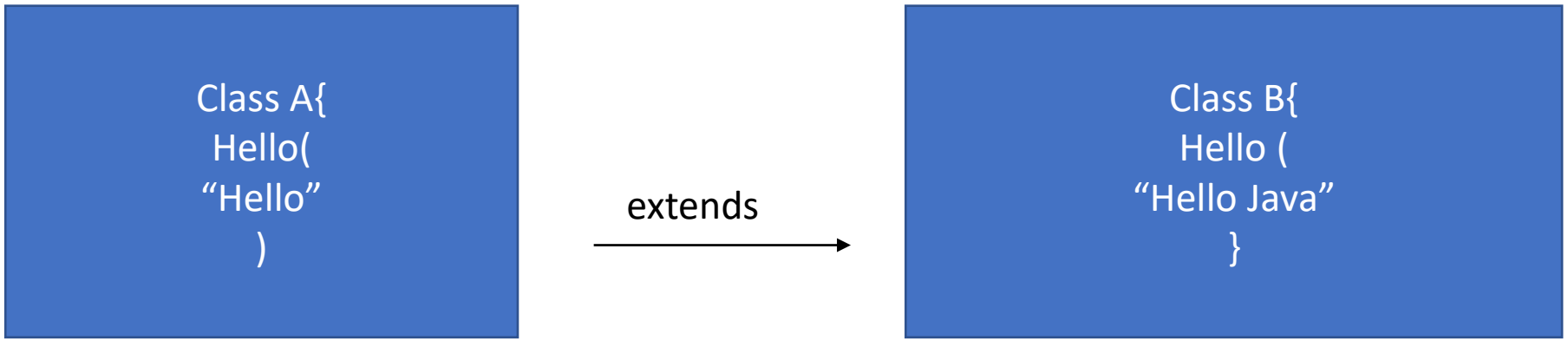
- RunTime Polymorphism:-

  When method declaration gets binded to its definition at the time of execution based on object or instance is called as runtime polymorphism.

  Method overriding is the example of runtime polymorphism. It is also called as late Binding.

  When the method body is changed in subclass after inheritance is known as method overriding.

  Points To Remember:- Always occur between two classes, and inheritance is a must for method overriding.



Method Overloading :-

Declaring multiple methods (Static or Non static) in a same class with same name but having different parameter or arguments, this concept is known as method overloading.

With the help of method Overloading we can declare multiple methods of same name, and can use this method of same name to perform different operations.

Method overloading is also known as compile time polymorphism.

For example: - Class Demo{

```
void test(){
Syso(“Zero Arguments”)
        }
void test(inta) {
Syso(“Int arguments”)
        }
```

Method Overriding:-

When one class acquires the property of another class i.e. when a subclass acquires the methods of superclass and in sub class its changes its implementation(Definition or method Body) according to sub class specification.

In method overriding method declaration is same but definition or body is different.

Ex:-
```
Class Demo{
    public void test(){
        Syso(“Hello”);
    }

    public void test1(){
        Syso(“Hi”)
    }
```

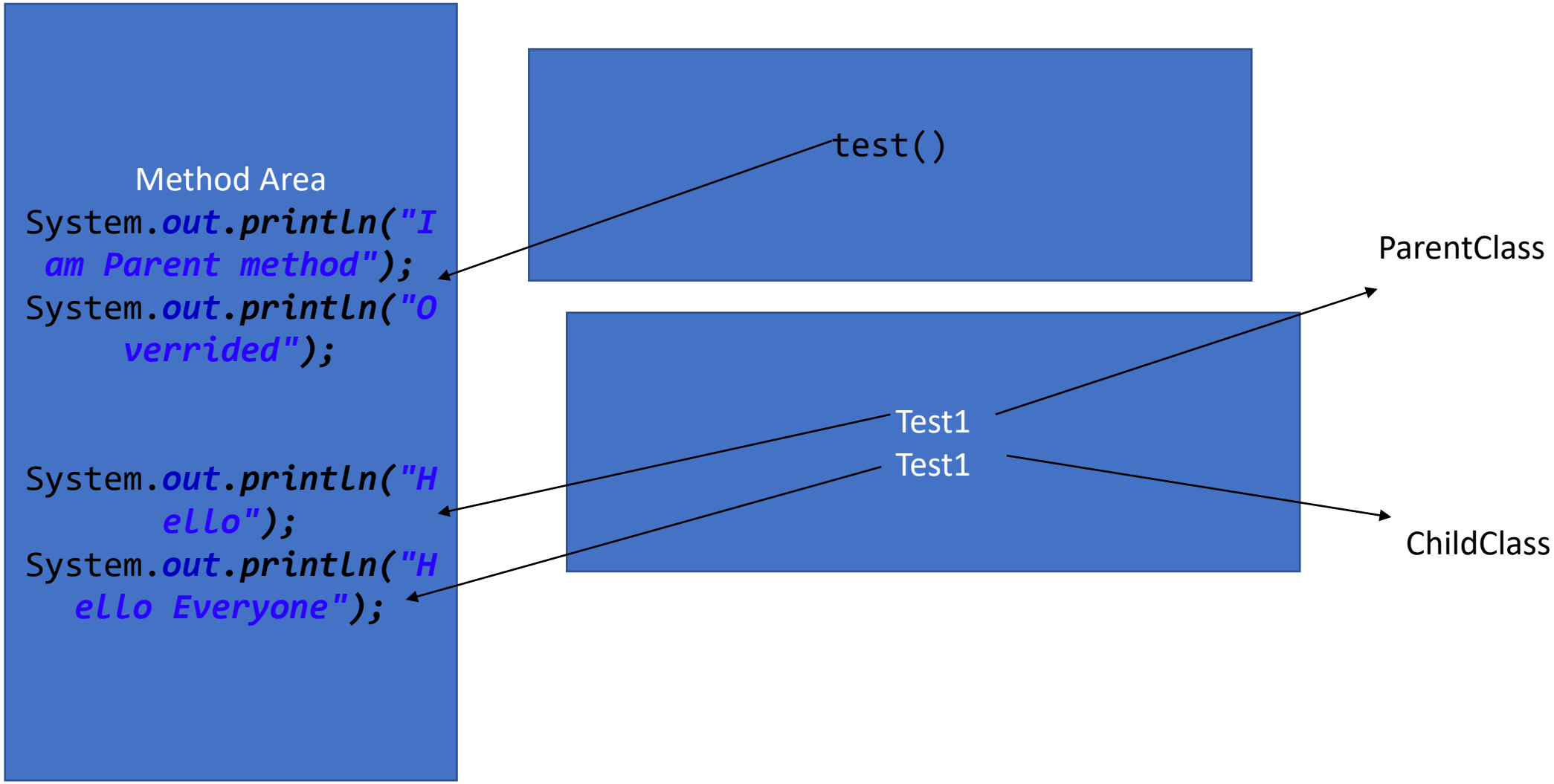
Ex:-
```
Class Demo2 extends Demo{
    public void test(){
        Syso(“Hello Java”);
    }
main {
 Demo2 obj = new Dome2();
 syso(obj.test);  //output :- Hello Java
 syso(obj.test1);//output :- Hi
}
```

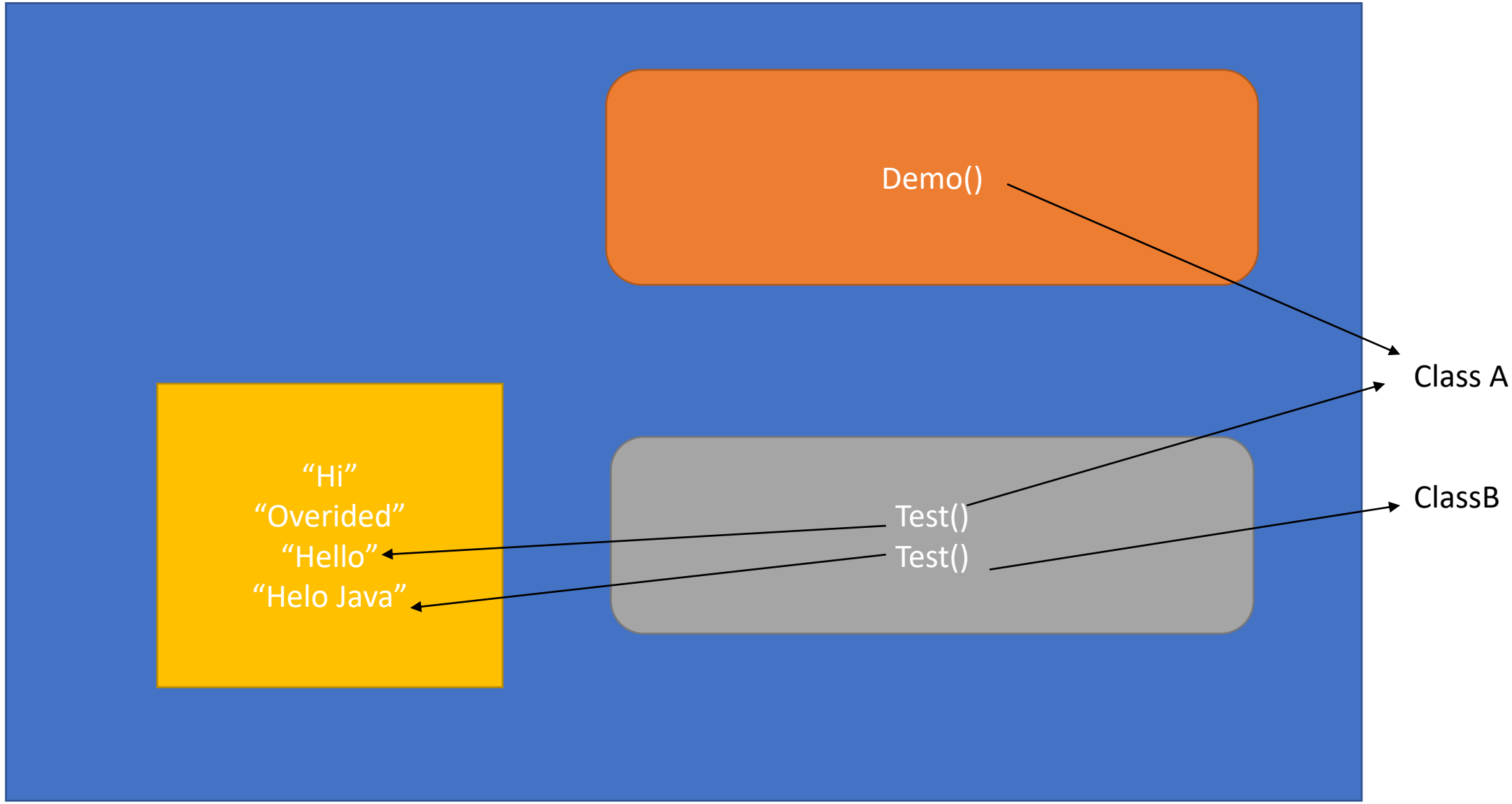
Super keyword:-

The super keyword refers to superclass (parent) objects.
It is used to call superclass methods, or it is used to call the members of superclass .The most common use of the super keyword is to eliminate the confusion between superclasses and subclasses that have methods with the same name.

This keyword:-

The `this` keyword refers to member of current class in which it is written.


Method Area
System.out.println("I am Parent method");
System.out.println("Overrided");
System.out.println("Hello");
System.out.println("Hello Everyone");
test()
Test1
Test1
ParentClass
ChildClass


Demo()
Class A
“Hi”
“Overided”
“Hello”
“Helo Java”
Test()
Test()
ClassB