CODING SHUTTLE

Week 5 LIVE

Java Memory, Polymorphism and Object Class

In This Lecture

1. How Java Memory Works
2. Java Object Class
3. Java Polymorphism

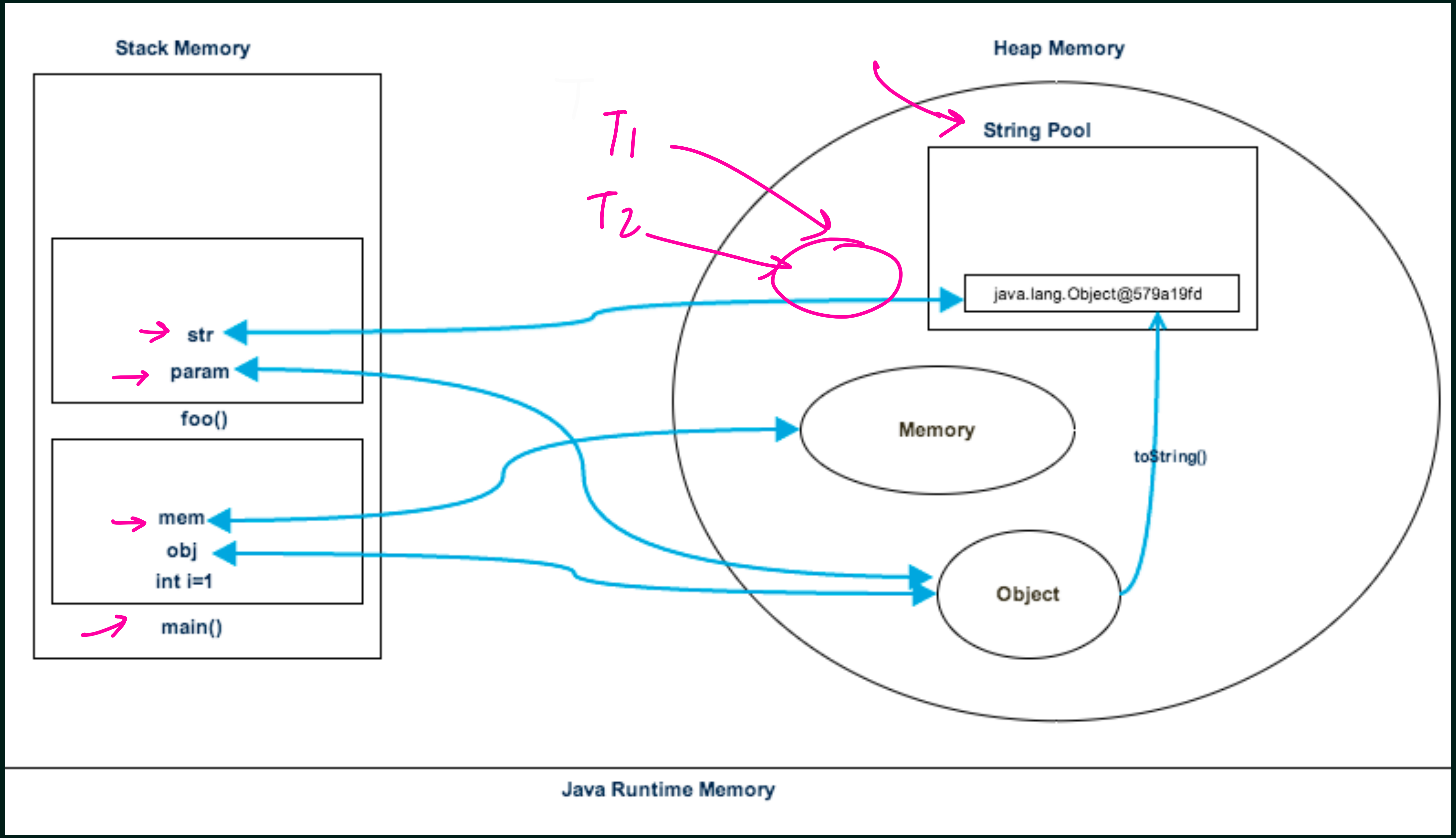
Java Heap And Stack Memory

↳ Garbage collector

```
public class Memory {
    int j = 2;
    public static void main(String[] args) { // Line 1
        int i=1; // Line 2
        Object obj = new Object(); // Line 3
        Memory mem = new Memory(); // Line 4
        mem.foo(obj); // Line 5
    } // Line 9

    private void foo(Object param) { // Line 6
        String str = param.toString(); //// Line 7
        System.out.println(str);
    } // Line 8

}
```



Java Heap Memory

Java Heap space is used by java runtime to allocate memory to Objects and JRE classes. Whenever we create an object, it's always created in the Heap space.

Any object created in the heap space has global access and can be referenced from anywhere of the application.

Java Stack Memory

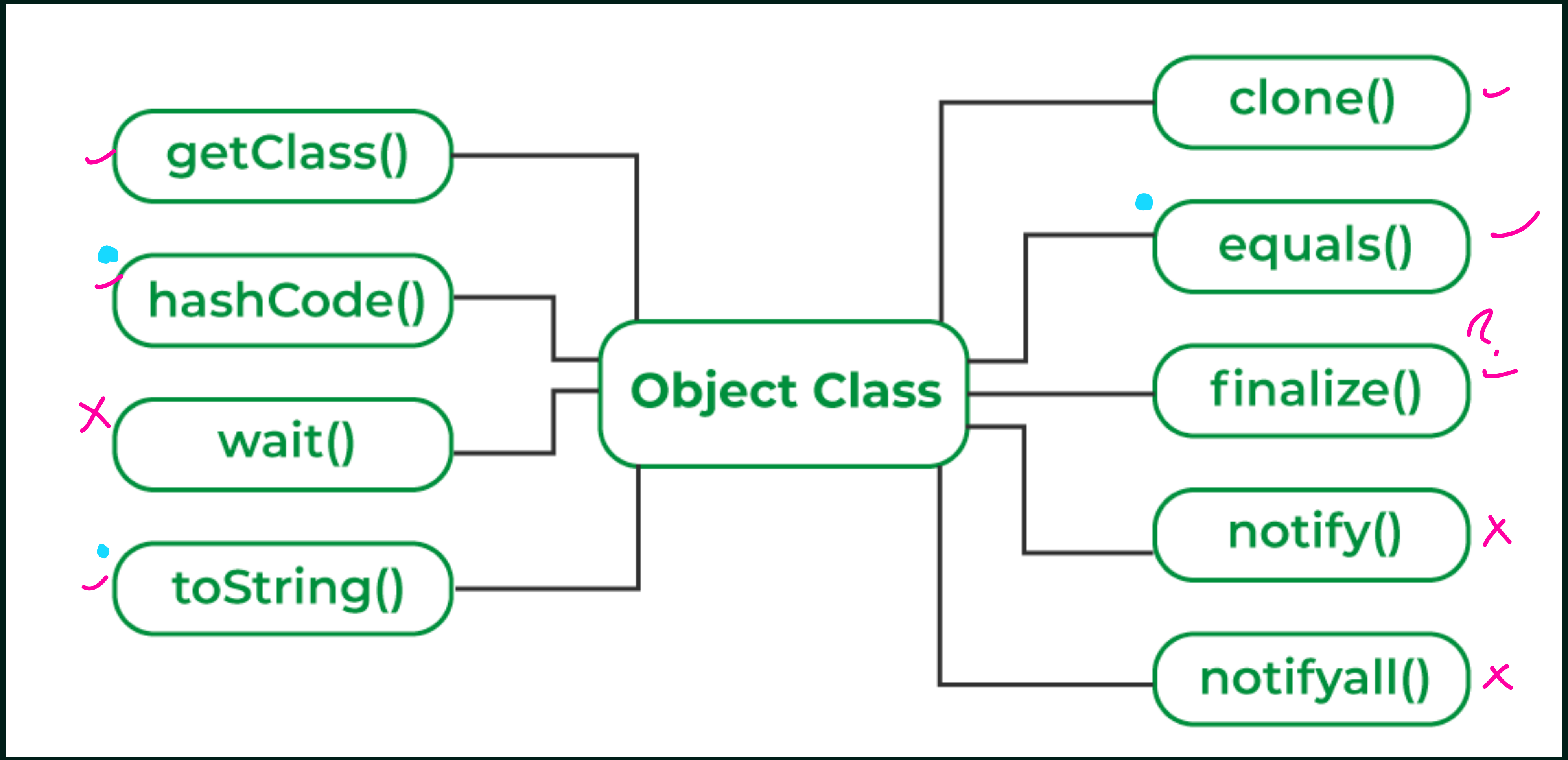
Java Stack memory contains method-specific values that are short-lived and references to other objects in the heap that is getting referred from the method.

Whenever a method is invoked, a new block is created in the stack memory for the method to hold local primitive values and reference to other objects in the method. As soon as the method ends, the block becomes unused and becomes available for the next method. Stack memory size is very less compared to Heap memory.

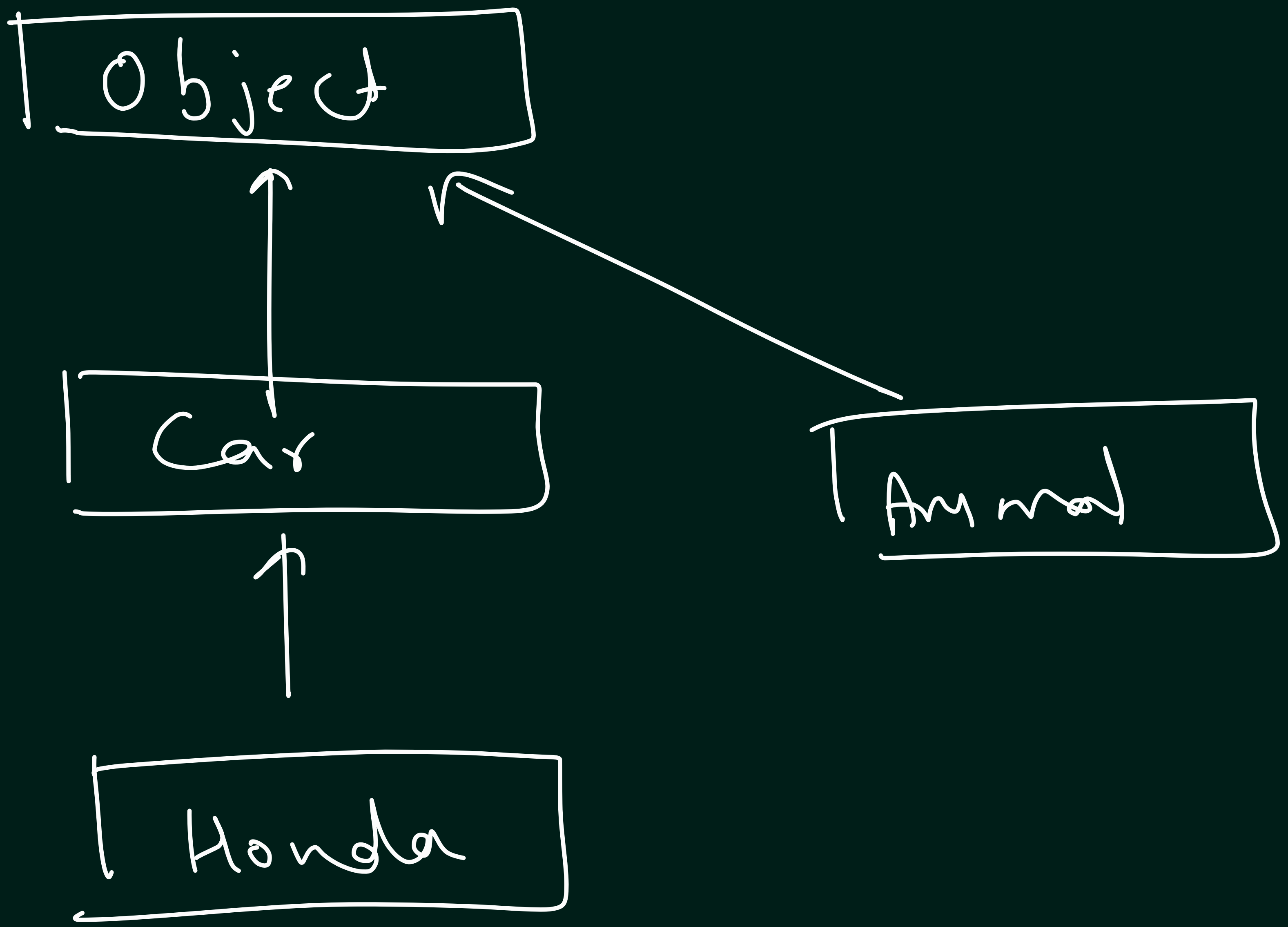
CODING SHUTTLE

Java Object Class

Object class is present in java.lang package. Every class in Java is directly or indirectly derived from the Object class. If a class does not extend any other class then it is a direct child class of Object and if extends another class then it is indirectly derived. Therefore the Object class methods are available to all Java classes. Hence Object class acts as a root of the inheritance hierarchy in any Java Program.



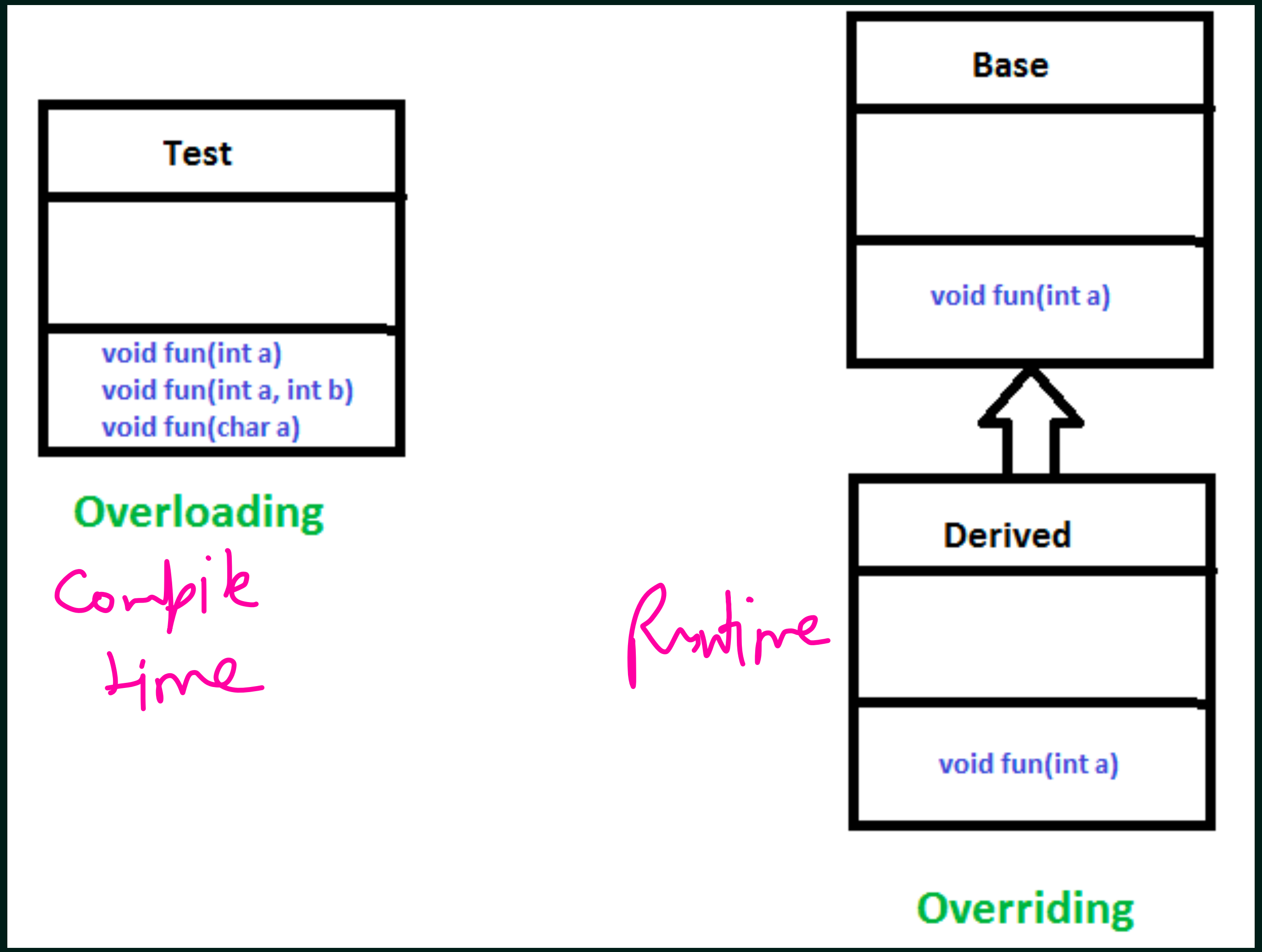
Java Object Class



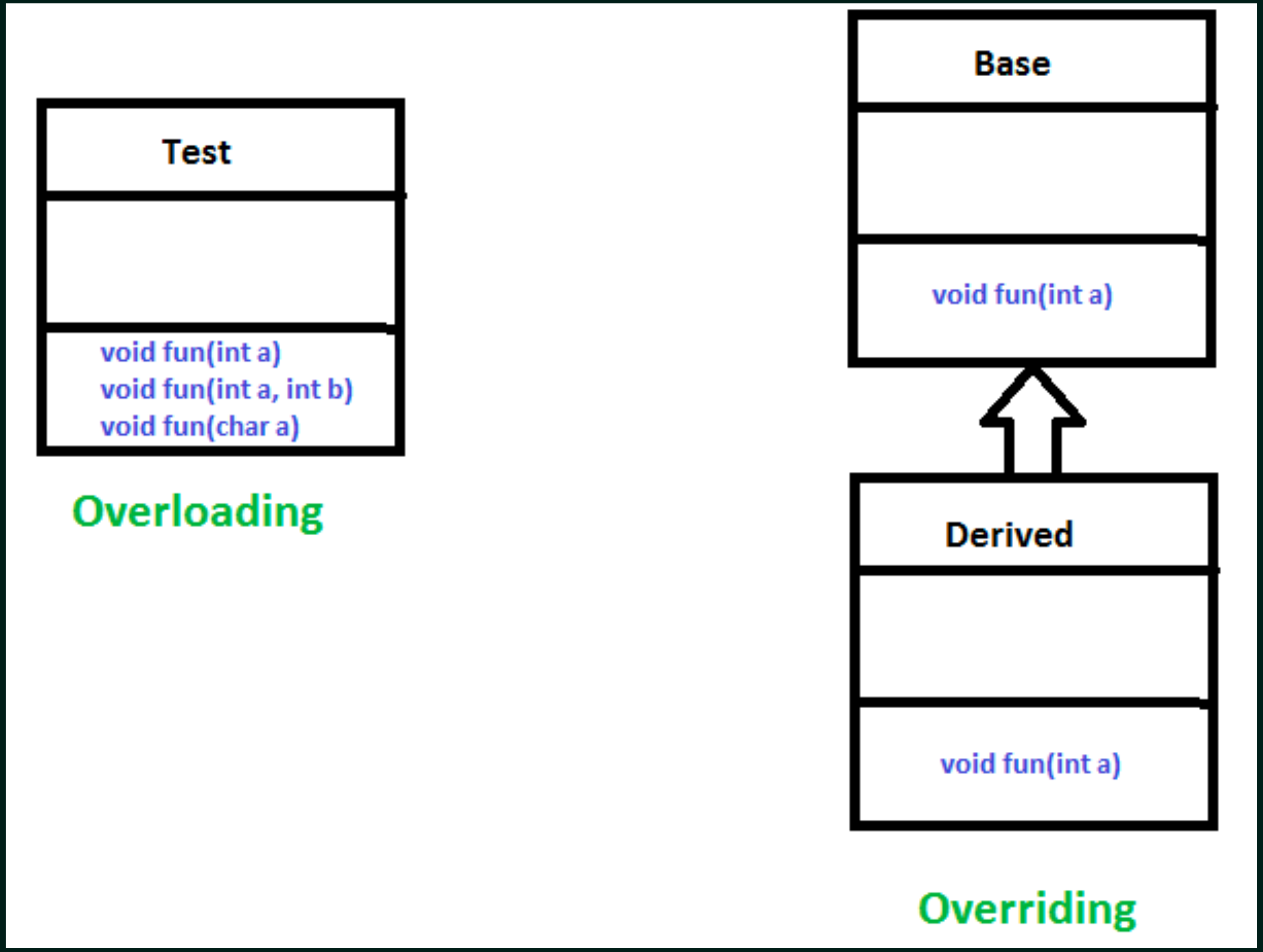
Java Polymorphism

Polymorphism allows us to perform a single action in different ways. In other words, polymorphism allows you to define one interface and have multiple implementations. The word "poly" means many and "morphs" means forms, So it means many forms. There are two types of Polymorphisms.

- Compile-time Polymorphism
- Runtime Polymorphism



Java Polymorphism



Test
void fun(int a)
void fun(int a, int b)
void fun(char a)
Overloading
Base
void fun(int a)
Derived
void fun(int a)
Overriding