As per problem, we have SuperDepartment class which is the super or base class of all other departments- AdminDepartment, TechDepartment and HrDepartment. According to this scenario, we have to apply the concept of inheritance and polymorphism to produce the desired output. The type of inheritance that we have to follow is Hierarchical inheritance and polymorphism is of type runtime polymorphism (dynamic binding).

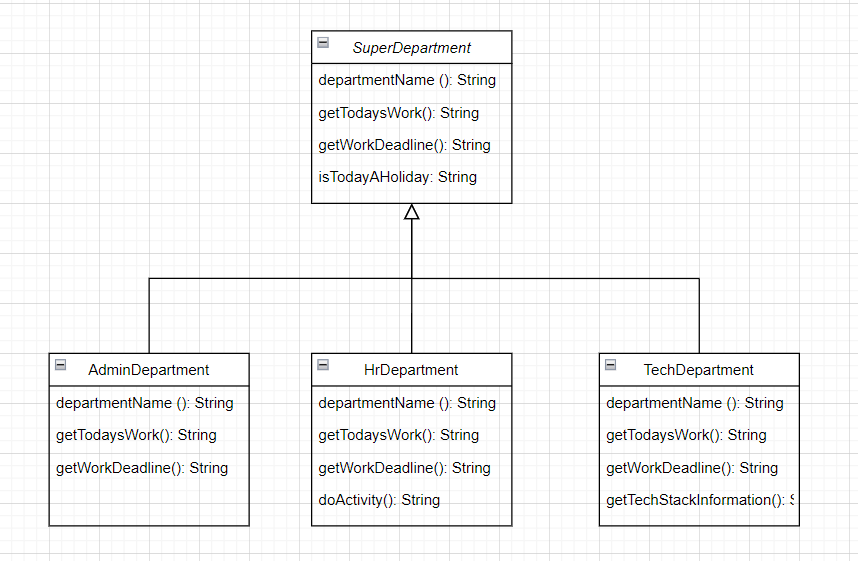


Figure 1: Hierarchical Inheritance

**Inheritance**

Inheritance is the process in which object of one class can link and share some common functionalities with the object of another class. We achieve inheritance by using the ‘extends’ keyword. The biggest advantage of inheritance is code re-usability.

There are five types of inheritance:

1. Single inheritance
2. Multi-level inheritance
3. Multiple inheritance (only through Interfaces)
4. Hierarchical inheritance: In this kind of inheritance, one class serves as a base class for more than one subclass. As per the given problem, SuperDepartment serves as the base class while AdminDepartment, HrDepartment and TechDepartment are subclasses of it. As we are going to apply this inheritance concept and so we are not going to cover the other inheritance types in this documentation.

In the demo example provided below, it is an example of Hierarchical inheritance.

1. Hybrid inheritance (only through interfaces)

Note: Java classes do not support multiple inheritance.

**Polymorphism**

The word ‘poly’ means multi and ‘morphic’ means shapes/behaviour/forms.

There are two types of polymorphism:

1. Compile-time polymorphism or static binding: We achieve this using method overloading. Method overloading is where we define multiple methods with exact same name but different signature. Here, the compiler links which specific method should be executed by JVM at runtime.

Since we will be not using method overloading in the problem, so we are only going to discuss about method overriding.

1. Runtime polymorphism or dynamic binding: We achieve this using method overriding. Method overriding is the process of a child class having the same exact method with the same exact method signature as that of a parent but with a different implementation. Runtime polymorphism is the process of holding the child classes functionality or child classes reference using the parent class reference and accessing the functionality of the child class using the parent class reference.

For example: A simple example to demonstrate hierarchical inheritance and runtime polymorphism.



Here in this program, when an object of child class is created, then the method inside the child class is called because the method in the parent class is overridden by the child class. Since the method is overridden, it has more priority than the parent method inside the child class.