OOP using Java

Trainer: Mr. Rohan Paramane

Singleton Design Pattern

- The singleton design pattern allows only to create one instance of your class.
- This is achieved by making the constructor private.
- As the constructor becomes private its instance cannot be created outside the class.
- The instance is created inside the class only once and same instance is returned every time through the static getter method.
- Requirements for singleton design patter
  - Private constructor
  - Static field of same type as that of class
  - Static Getter method for the field to return its instance.

Hireracy

- It is a major pillar of oops.
- Level / order / ranking of abstraction is called hierarchy.
- Main purpose of hierarchy is to achieve reusability.
- Advantages of code reusability
    1. We can reduce development time.
    2. We can reduce development cost.
    3. We can reduce developers effort.
- Types of hierarchy:
    1. Has-a / Part-of => Association
    2. Is-a / Kind-of => Inheritance /
    3. Use-a => Dependency
    4. Creates-a => Instantiation

Association

- If has-a relationship association is exist between the types then we should use
- Example
  1. Car has a engine
  2. Room has a chair

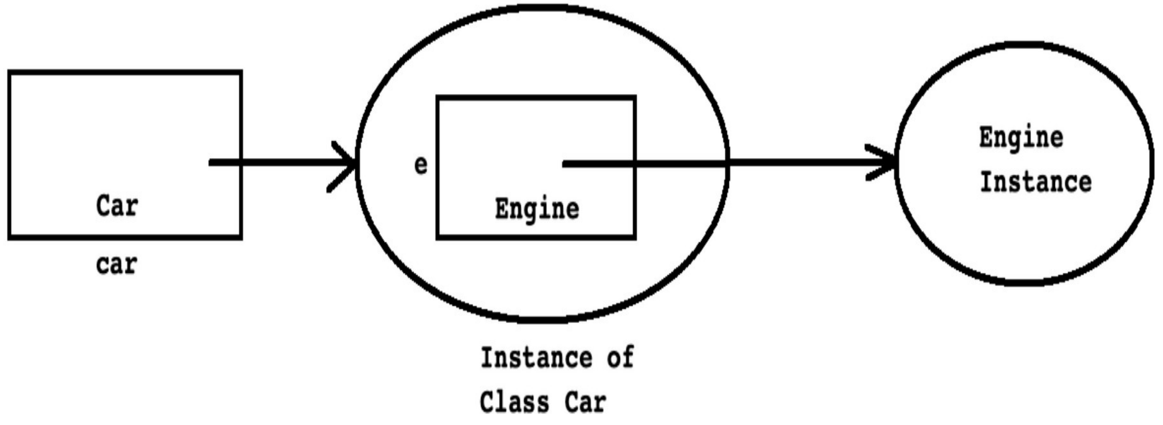
- Let us consider example of car and engine:
  - Car has a engine
  - Engine is part of Car.

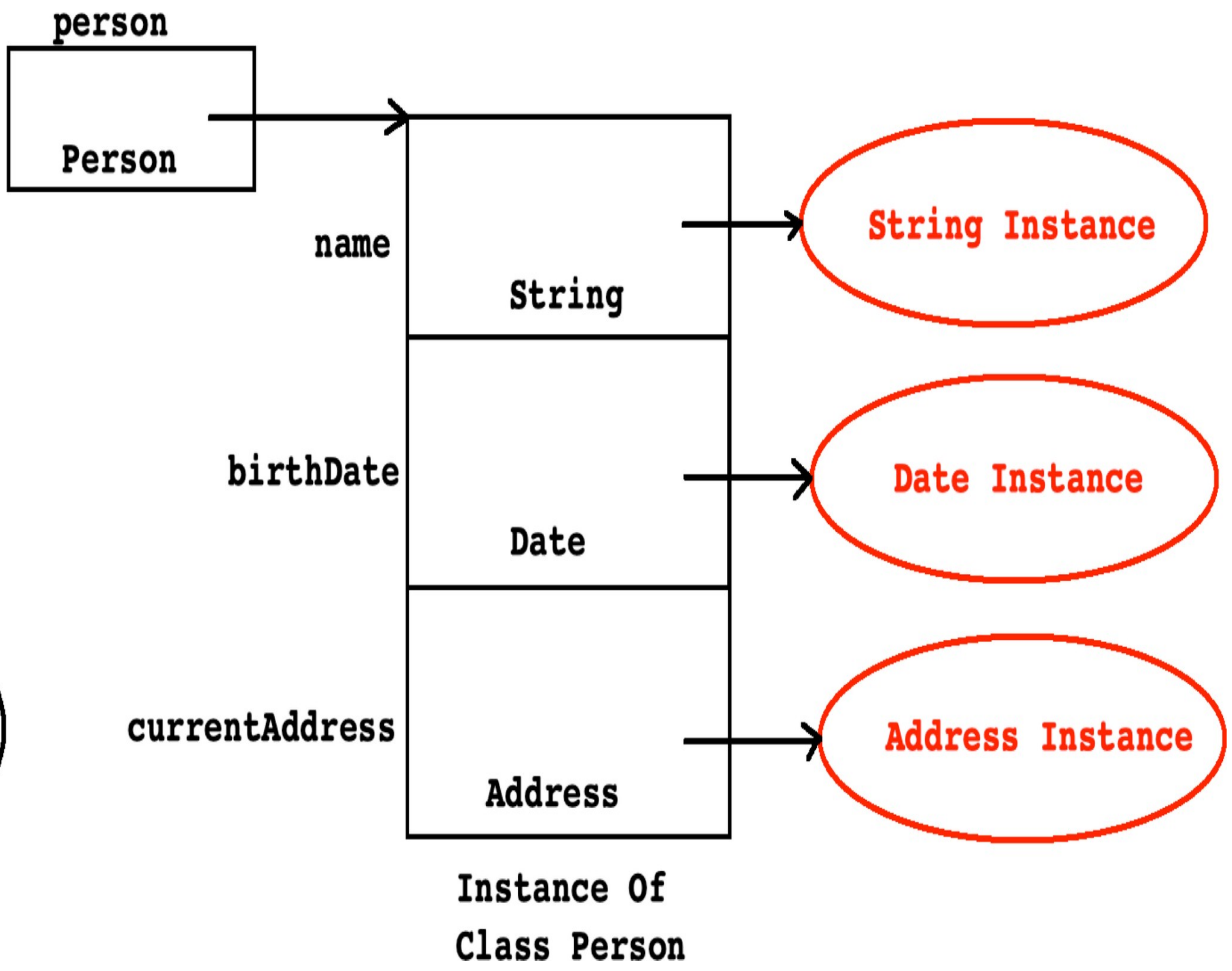
- If object/instance is a part/component of another instance then it is called as association.
- To implement association, we should declare instance of a class as a field inside another class.

Association

- Engine is part of Car

```
class Engine{
  //TODO
}
class Car{
  Engine e = new Engine( ); //Association
}
Car c = new Car( );
```





Inheritance

- If "is-a" relationship is exist between the types then we should use inheritance.
- Inheritance is also called as generalization.
- Example
    1. Manager is a employee
    2. Book is a product
    3. Triangle is a shape
    4. SavingAccount is a account.

```
class Employee{ //Parent class
    //TODO
}
class Manager extends Employee{ //Child class
    //TODO
}
//Here class Manager is extended from class Employee.
```

- If we want to implement inheritance then we should use extends keyword.
- In Java, parent class is called as super class and child class is called as sub class.
- Java do not support private and protected mode of inheritance
- In Java, class can extend only one class.
- In other words, multiple class inheritance is not allowed.
- Consider following code:

```
class A{    }
class B{    }
class C extends A, B{   //Not OK
}
```

Inheritance

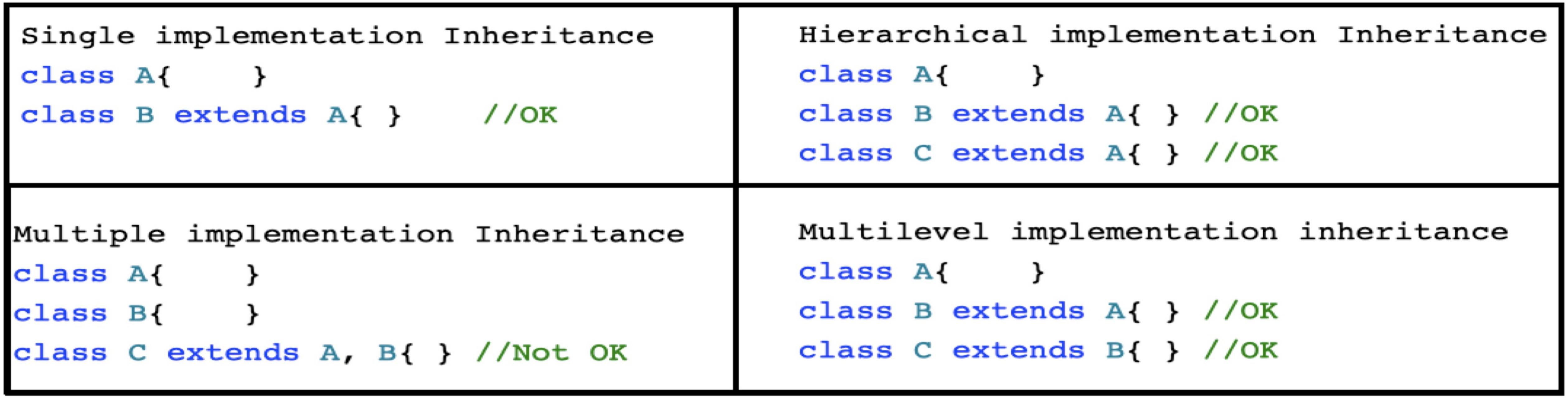
- If we create instance of sub class then all the non static fields declared in super class and sub class get space inside it. In other words, non static fields of super class inherit into sub class.
- Static field do not get space inside instance. It is designed to share among all the instances of same class.
- Using sub class, we can access static fields declared in super class. In other words, static fields of super class inherit into sub class.
- All the fields of super class inherit into sub class but only non static fields gets space inside instance of sub class.
- Fields of sub class, do not inherit into super class. Hence if we create instance of super class then only non static fields declared in super class get space inside it.
- If we declare field in super class static then, all the instances of super class and sub class share single copy of static field declared in super class.
- We can call/invoke, non static method of super class on instance of sub class. In other words, non static method inherit into sub class.
- We can call static method of super class on sub class. In other words, static method inherit into sub class.
- Except constructor, all the methods of super class inherit into sub class.

Inheritance

- If we create instance of super class then only super class constructor gets called.
- But if we create instance of sub class then then JVM first give call to the super class constructor and then sub class constructor.
- From any constructor of sub class, by default, super class's parameterless constructor gets called.
- Using super statement, we can call any constructor of super class from constructor of sub class.
- Super statement, must be first statement inside constructor body.
- According to client's requirement, if implementation of super class is logically incomplete / partially complete then we should extend the class. In other words we should use inheritance.
- According to client's requirement, if implementation of super class method is logically incomplete / partially complete then we should redefine method inside sub class.
- Process of redefining method of super class inside sub class is called as **method overriding.**

Types of Class Inheritance

- During inheritance, if super type and sub type is class, then it is called as implementation inheritance.
- Types of inheritance
  1. Single Implementation Inheritance
  2. Multilevel Implementation Inheritance
  3. Hierarchical Implementation Inheritance



```
Single implementation Inheritance
class A{     }
class B extends A{ }     //OK
```

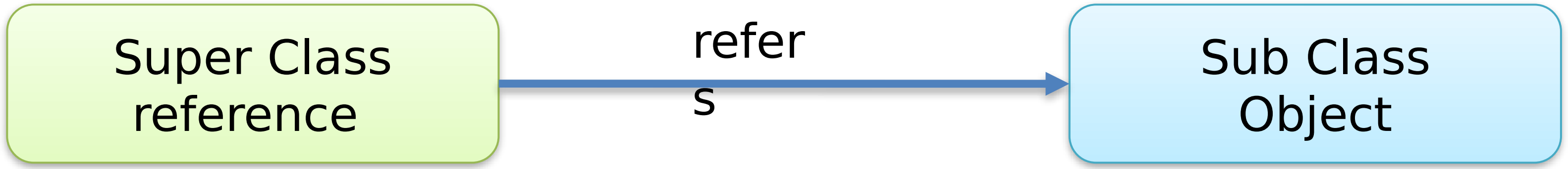
```
Hierarchical implementation Inheritance
class A{     }
class B extends A{ } //OK
class C extends A{ } //OK
```

```
Multiple implementation Inheritance
class A{     }
class B{     }
class C extends A, B{ } //Not OK
```

```
Multilevel implementation inheritance
class A{     }
class B extends A{ } //OK
class C extends B{ } //OK
```

Upcasting

- When the reference variable of super class refers to the object of subclass, it is known as upcasting in java.
- when subclass object type is converted into superclass type upcasting.

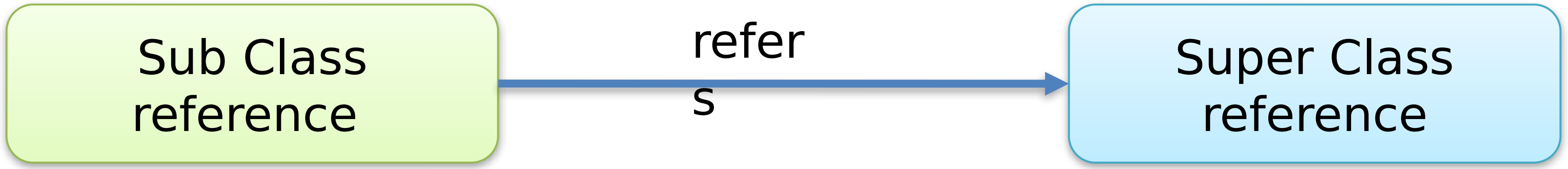


Superclass s = new SubClass();

- Upcasting gives us the flexibility to access the parent class members, but it is not possible to access all the child class members using this feature.
- Instead of all the members, we can access some specified members of the child class.
- For instance, we can only access the overridden methods in the child class.

Downcasting

- Assigning parent class reference (which is pointing to child class object) to child class reference .
- Syntax for down casting : Child c = (Child)p;
- Here p is pointing to the object of child class as we saw earlier and now we cast this parent reference p to child class reference c.
- Now this child class reference c can access all the methods and variables of child class as well as parent class.



Subclass sub = (Subclass) s ;

Dynamic Method Dispatch

- Process of redefining method of super class inside sub class is called as **method overriding.**
- When we keep instance of subclass in superclass reference we call it as **upcasting.**
- When such super class ref is used to invoke the overriding method, then the decision to send the method for execution is taken by JRE & not by compiler.
- In such case overriding form of the method(sub-class version) will be dispatched for execution.
- This is called as **Dynamic Method Dispatch.**
- Javac resolves the method binding by the type of the reference & JVM resolves the method binding by type of the object it's referring to.

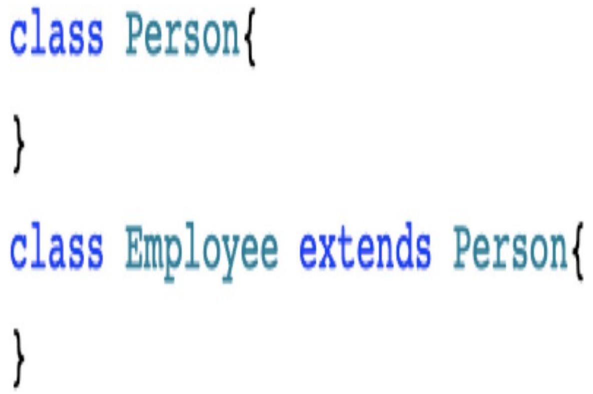
Instanceof

- Downcasting is not always safe, as we explicitly write the class names before doing downcasting.
- It won't give an error at compile time but it may throw ClassCastExcpetion at run time, if the parent class reference is not pointing to the appropriate child class.
- To remove ClassCastException we can use instanceof operator to check right type of class reference in case of down casting .

```
If (s instance of Subclass)  //(<Superclass_Ref> instanceof <subclass_className>)
{
Subclass sub = (Subclass) s;
}
```

Object class

- It is a non final and concrete class declared in java.lang package.
- In java all the classes(not interfaces)are directly or indirectly extended from java.lang.Object class.
- In other words, java.lang.Object class is ultimate base class/super
- cosmic base class/root of Java class hierarchy
- Object class do not extend any class or implement any interface.
- It doesn’t contain nested type as well as field.
- It contains default constructor.
  - Object o = new Object(“Hello”); // NOT OK
  - Object o = new Object( ); // OK
- Object class contains 11 methods.
- In the given code, java.lang.object is direct super class of class Person.
- For class Employee, class Person is direct super class and class object is indirect super class.



```
class Person{
}
class Employee extends Person{
}
```

toString( ) method

- It is a non final method of java.lang.Object class.
- Syntax:
    - public String toString( );
- If we want to return state of Java instance in String form then we should use toString() method.
- If we do not define toString() method inside class then super class's
-   toString() method gets call.
- If we do not define toString() method inside any class then object class's
- toString() method gets call.
- It return String in the form: F.Q.ClassName@HashCode
    - Example : test.Employee@6d06d69c
- If we want state of instance then we should override toString() method inside class.
- The result in toString method should be a concise but informative that is easy for a person to read.
- It is recommended that all subclasses override this method.

SUNBEAM

Thank you!

Rohan Paramane

rohan.paramane@sunbeaminfo.com