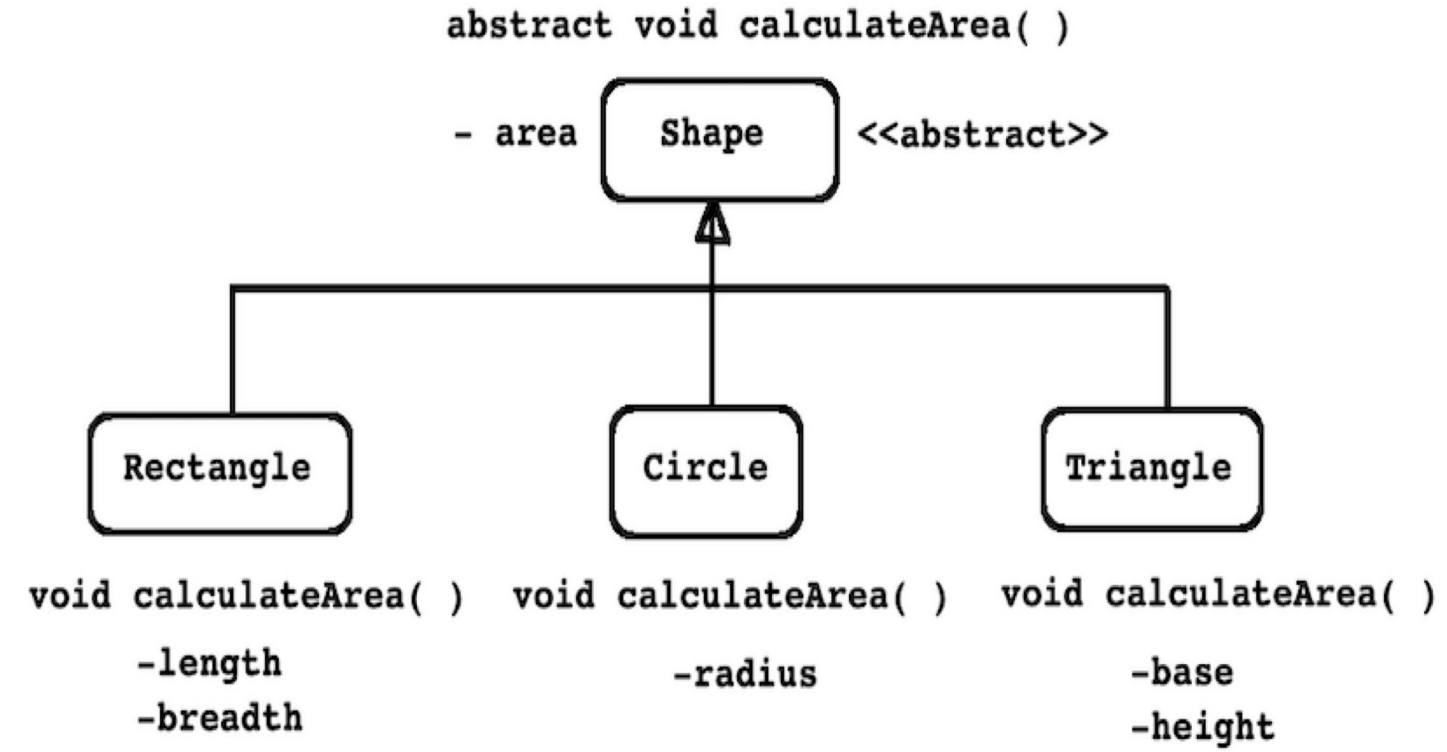
OOP using Java

Trainer: Mr. Rohan Paramane

Abstract Class

- If "is-a" relationship is exist between super type and sub type and if we want same method design in all the sub types then super type must be abstract.
- Using abstract class, we can group instances of related type together
- Abstract class can extend only one abstract/concrete class.
- We can define constructor inside abstract class.
- Abstract class may or may not contain abstract method.
- **Hint :** In case of inheritance if state is involved in super type then it should be abstract.




```
Shape[] arr = new Shape[ 3 ];
arr[ 0 ] = new Rectangle( ); //Upcasting
arr[ 1 ] = new Circle( ); //Upcasting
arr[ 2 ] = new Triangle( ); //Upcasting
```

Interface

- In Java, an interface is a blueprint or template of a class. It is much similar to the Java class but the only difference is that it has abstract methods and static constants.
- The methods in interfaces do not contain any body.
- An interface in Java is a mechanism which we mainly use to achieve abstraction and multiple inheritances in Java.
- An interface provides a set of specifications that other classes must implement.
- We can implement multiple Java Interfaces by a Java class.
- All methods of an interface are implicitly public and abstract.
- The word abstract means these methods have no method body, only method signature.
- An interface can inherit or extend multiple interfaces.
- We can implement more than one interface in our class.

Interface

- If"is-a" relationship is not exist between super type and sub type and if we want same method design in all the sub types then super type must interface.
- Using interface, we can group instances of unrelated type together.
- Interface can extend more than one interfaces.
- We can not define constructor inside interface.
- By default methods of interface are abstract.
- **Hint :** In case of inheritance if state is not involved in super type then it should be interface.
- Unlike a class, you cannot instantiate or create an object of an interface.
- All the methods in an interface should be declared as abstract.
- An interface does not contain any constructors, but a class can.
- An interface cannot contain instance fields.
- It can only contain the fields that are declared as both static and final.
- An interface can not be extended or inherited by a class; it is implemented by a class.
- An interface cannot implement any class or another interface.

Interface

- Set of rules are called specification/standard.
- It is a contract between service consumer and service provider.
- If we want to define specification for the sub classes then we should define interface.
- Interface is non primitive type which helps developer:
- To build/develop trust between service provider and service consumer.
- To minimize vendor dependency.
- interface is a keyword in Java.
- If we want to implement rules of interface then we should use implements keyword.
- It is mandatory to override, all the abstract methods of interface otherwise sub class can be considered as abstract.

Interface

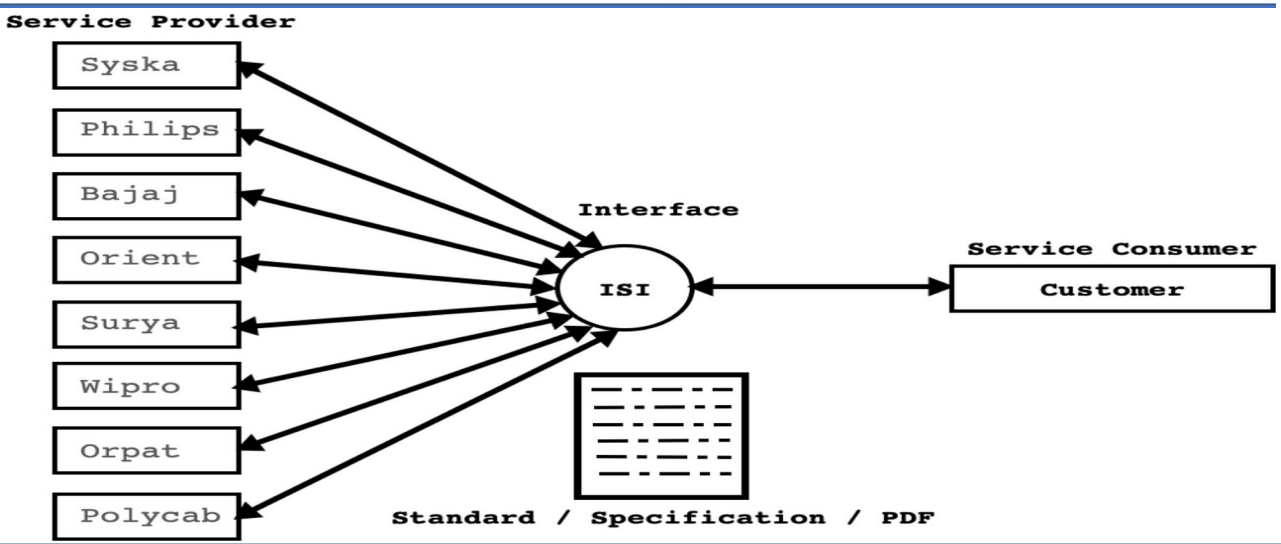
```
interface Printable{
    int number = 10;
    void print( );
}
```

* Solution 1

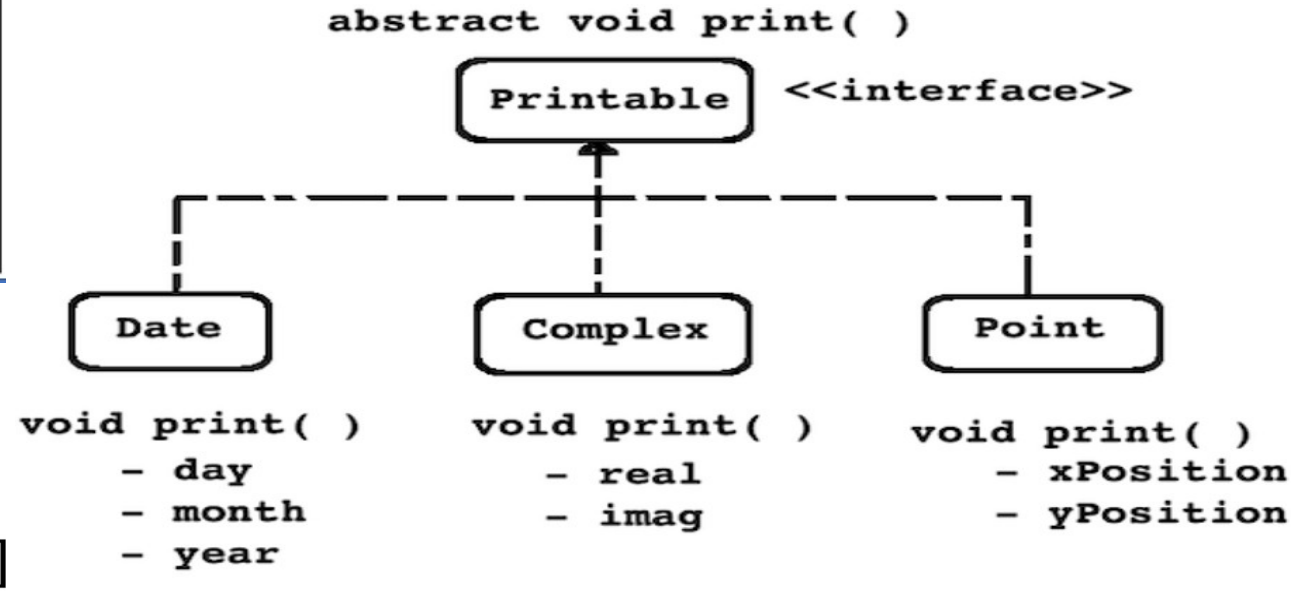
```
abstract class Test implements Printable{
}
```

* Solution 2

```
class Test implements Printable{
    @Override
    public void print( ){
        //TODO
    }
}
```



- If we want to implement rules of interface then we should use implements keyword.
- It is mandatory to override, all the abstract methods of interface otherwise sub class can be considered as abstract.




```
Printable[] arr = new Printable[ 3 ];
arr[ 0 ] = new Date( ); //Upcasting
arr[ 1 ] = new Complex( ); //Upcasting
arr[ 2 ] = new Point( ); //Upcasting
```

Types of Interface Inheritance

- During inheritance if super type and sub type is interface then it is called as interface inheritance.
- Types of interface inheritance
  1. Single Inheritance
  2. Multiple Inheritance
  3. Hierarchical Inheritance
  4. Multilevel Inheritance

Interface : I1, I2, I3
Class : C1, C2, C3

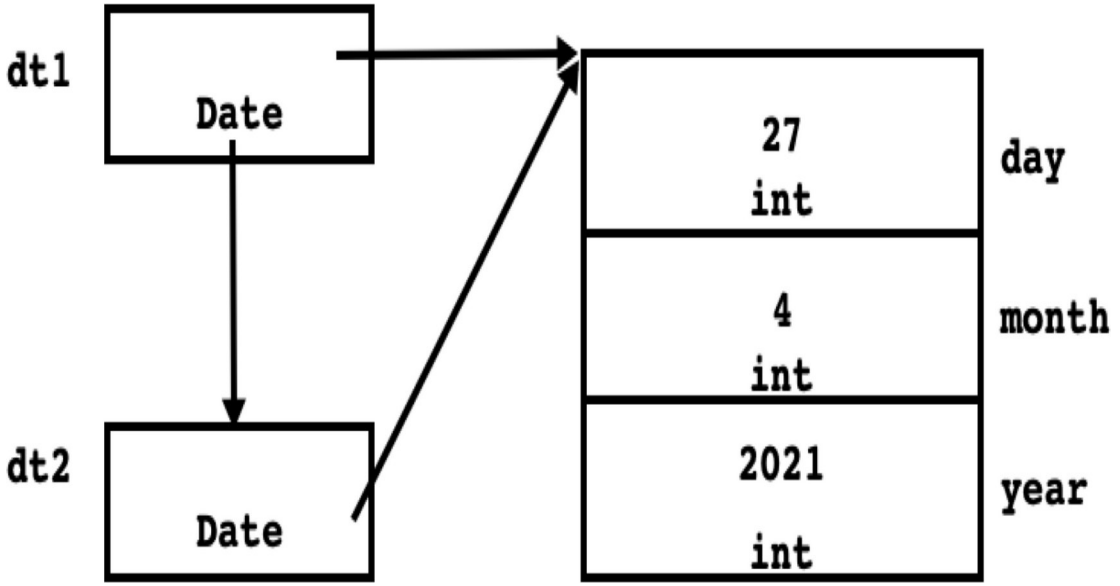
* I2 implements I1 //Incorrect
* I2 extends I1 //correct : Interface inheritance
* I3 extends I1, I2 //correct : Multiple interface inheritance
* C2 implements C1 //Incorrect
* C2 extends C1 //correct : Implementation Inheritance
* C3 extends C1,C2 //Incorrect : Multiple Implementation Inheritance
* I1 extends C1 //Incorrect
* I1 implements C1 //Incorrect
* c1 implements I1 //correct : Interface implementation inheritance
* c1 implements I1,I2 //correct : Multiple Interface implementation inheritance
* c2 implements I1,I2 extends C1 //Incorrect
* c2 extends C1 implements I1,I2 //correct

Marker Interface

- An interface which do not contain any member is called marker interface. In other words, empty interface is called as marker interface.
- Marker interface is also called as tagging interface.
- If we implement marker interface then Java compiler generates metadata for the JVM, which help JVM to clone/serialize or marshal state of object.
- Example:
  - java.lang.Cloneable
  - java.util.EventListener
  - java.util.RandomAccess
  - java.io.Serializable
  - java.rmi.Remote

Cloneable Interface Implementation

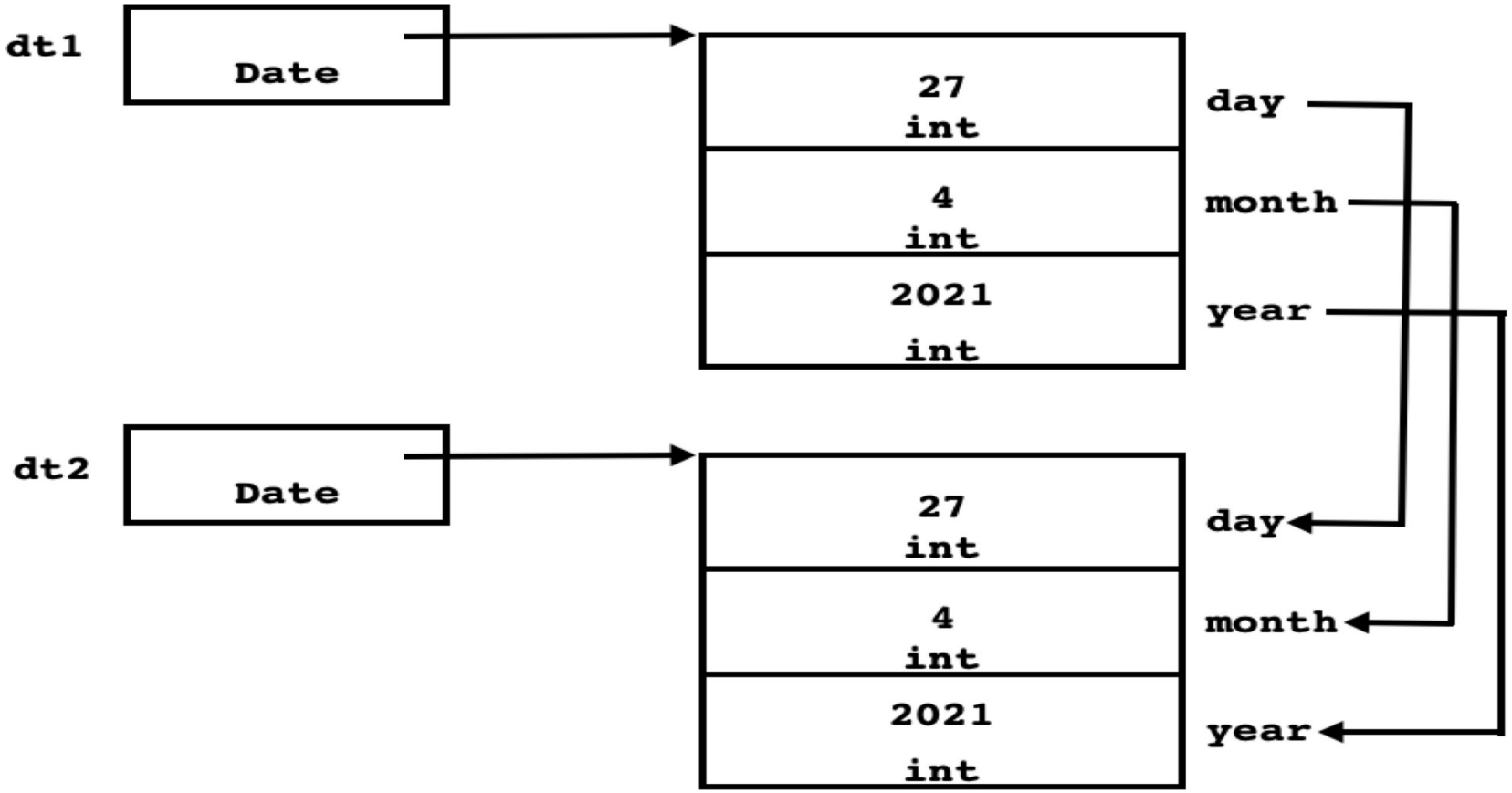
- Date dt1 = new Date(27, 4, 2021);
- Date dt2 = dt1; //Shallow Copy Of References



- If we want to create new instance from existing instance then we should use clone method.
- clone( ) is non final native method of java.lang.Object class.
- Inside clone() method, if we want to create shallow copy instance then we should use super.clone( ) method.
- Cloneable is interface declared in java.lang package.
- Without implementing Cloneable interface, if we try to create clone of the instance then clone() method throws CloneNotSupportedException.

Cloneable Interface Implementation

- Date dt1 = new Date(27, 4, 2021);
- Date dt2 = dt1.clone( ); //Shallow Copy Of Instance



Thank you!

Rohan Paramane

rohan.paramane@sunbeaminfo.com