Object orientated programming

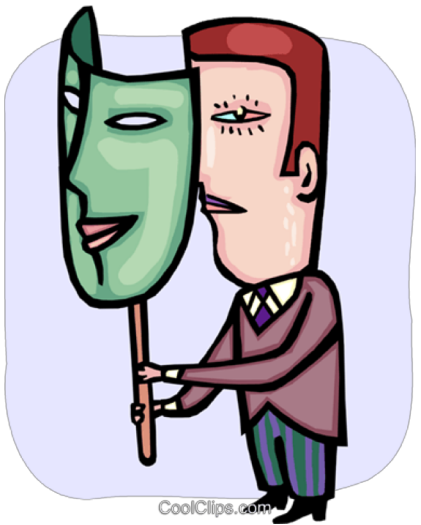
Abstract classes and interfaces

Abstract classes

- A class which is declared with the **abstract keyword** is known as an abstract class.
- An abstract class can have abstract and non-abstract methods.

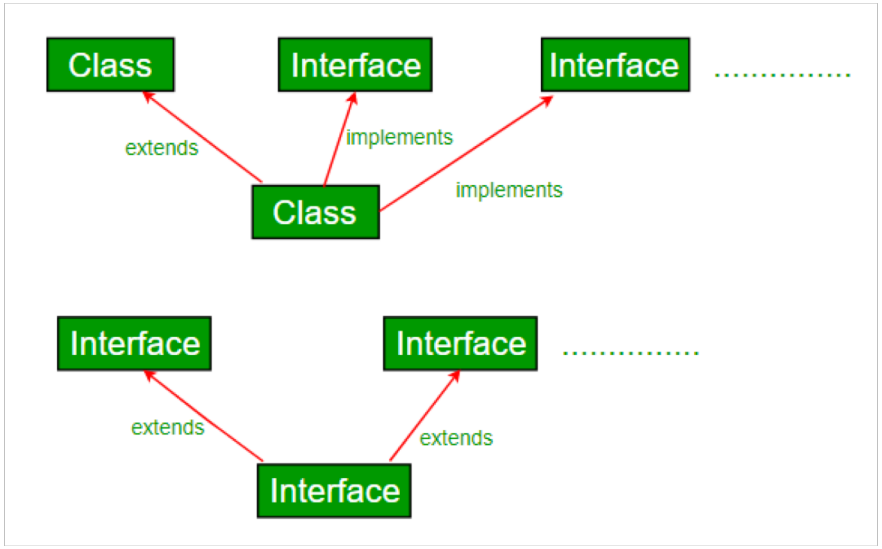
What is abstraction?

- **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
- **Showing only essential parts** of your program to the user and **hide** the internal **details**.
  - For example, sending a SMS where you type the text and send the message. The user does not know the internal processes how the message is delivered.
- Abstraction leads the **focus on what the object does instead of how** it does it.



How do we achieve abstraction?

- There are basically two ways
- **Abstract classes**
  - Can model abstraction on different levels (0-100%)
- **Interfaces**
  - Are 100% abstract



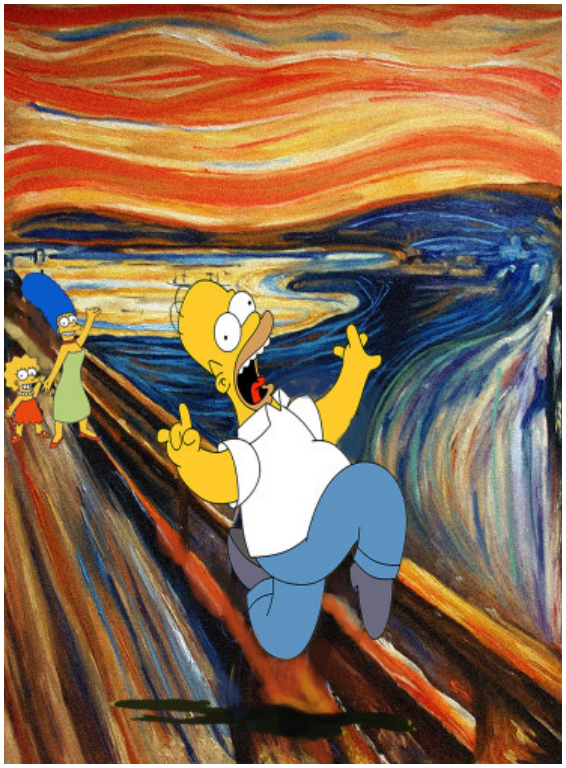
Abstract class

- As heard before, a class which is declared as abstract is known as an **abstract class**.
- Abstract classes come with some rules
  - It can consist of abstract and non-abstract methods.
  - It needs to be extended and the abstract methods need to be implemented.
  - It cannot be instantiated as an object.

```
abstract class A{}
```

Abstract classes – most important points

- An abstract class **must be declared** with an **abstract** keyword.
- It can have **abstract and non-abstract** methods.
- It **cannot** be **instantiated**.
- It **can** have **constructors** and **static methods**.
- It **can** have **final** methods
  - Which will force the subclass not to change the body of the method.



Abstract methods

- A method which is declared as abstract and does not have an implementation in the method body is known as an abstract method.

```
abstract void printStatus();//no method body and abstract
```

- Lets see an example (Bike)

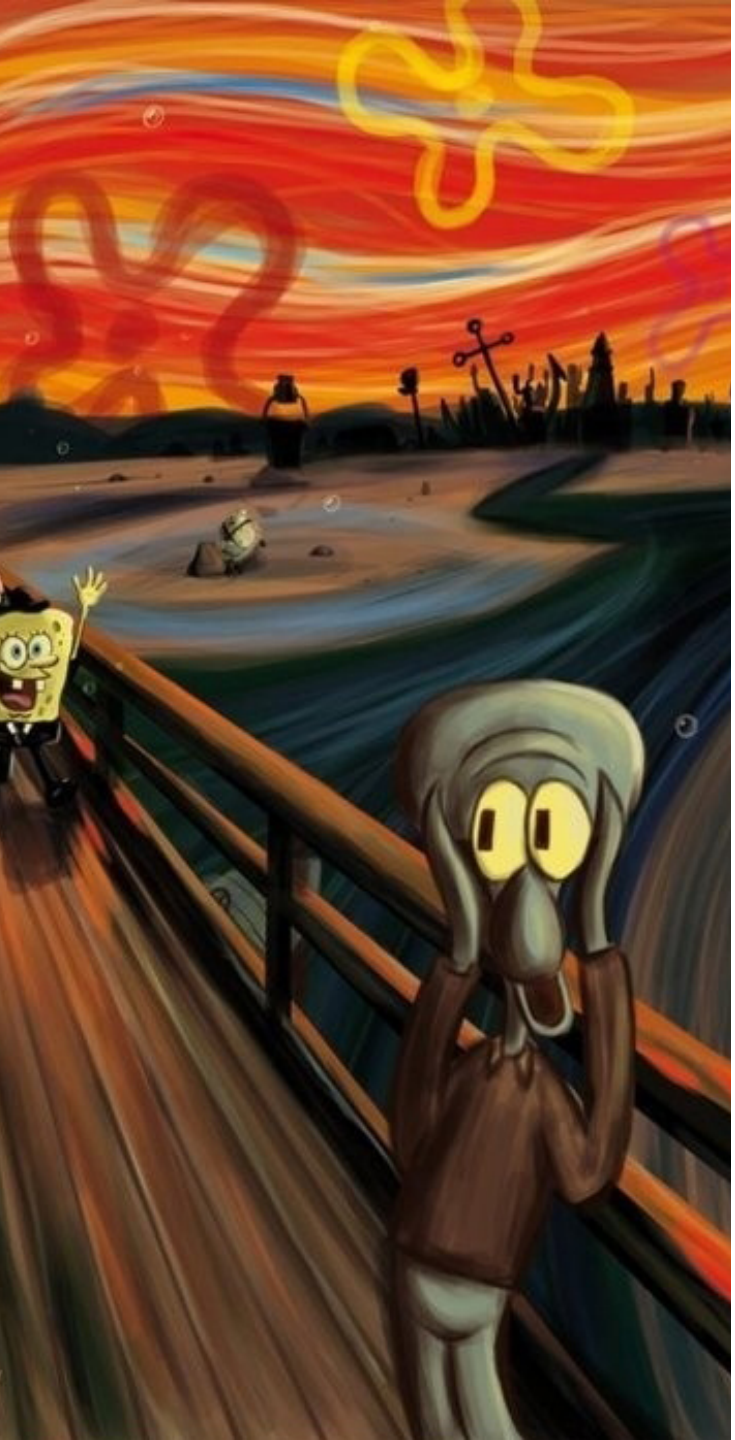
Better understanding by a real example

- Shape is the abstract class, and its implementation is provided by the Rectangl
- Mostly, v
(which is
class is p
- A **factory**
- In this example, if you create the instance of the Rectangle class, the draw() method of the Rectangle class will be invoked (TestAbstraction1).
- **Second example:** A banking system returning rates for different banks (TestBank).

"Define an interface for creating an object, but let subclasses decide which class to instantiate. The Factory method lets a class defer instantiation it uses to subclasses." (***Design Patterns: Elements of Reusable Object-Oriented Software*** (1994))

Abstract classes - extended

- An abstract class can have data members, abstract methods, method body (non-abstract method), constructors, and even a main() method.
- Example: Extension of our bike example with abstract and non abstract methods (TestAbstraction2).
- **Tipp: If there is an abstract method in a class, the class must be abstract (otherwise you get compile time error).**
- **Tipp: If you are extending an abstract class that has an abstract method, you must either provide the implementation of the method or make the extended class abstract again.**

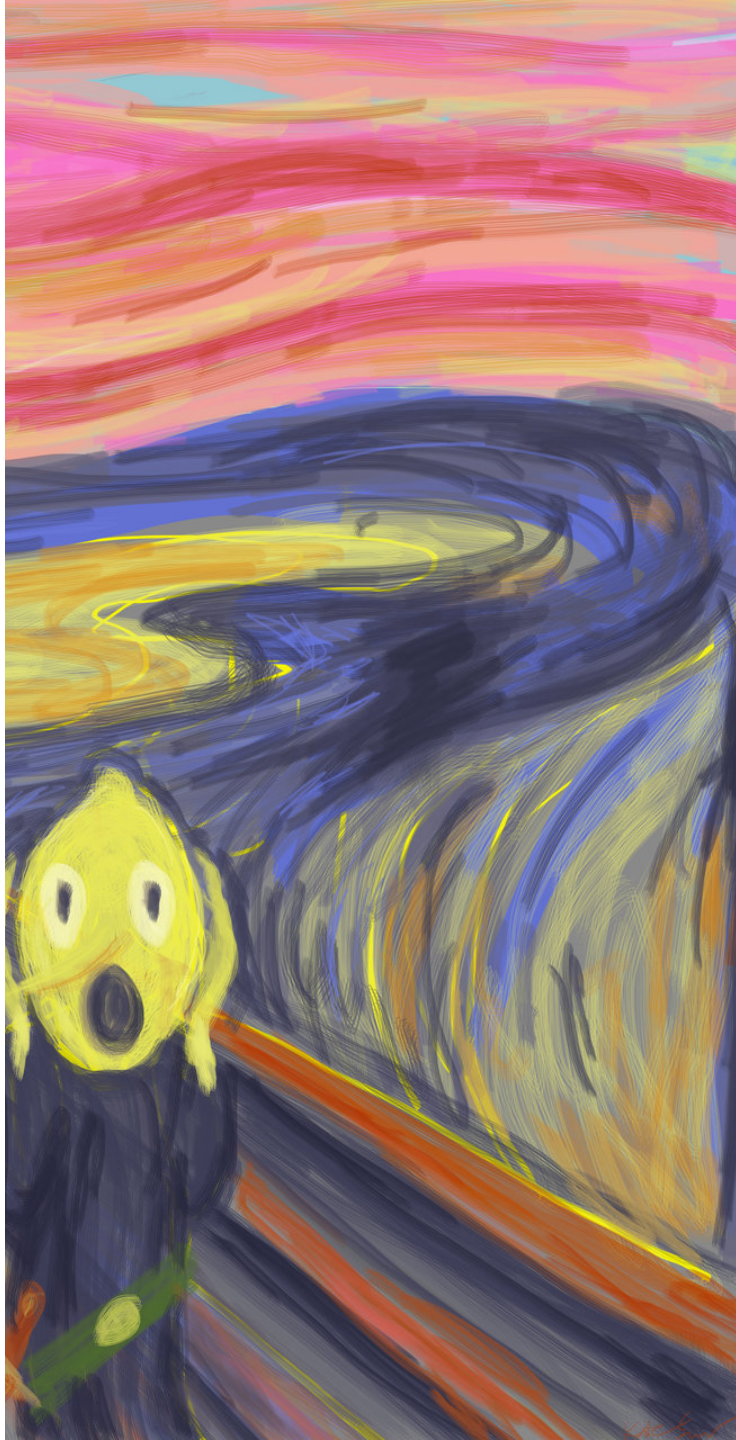


Interfaces (i)

- An **interface in java** is a blueprint of a class (blueprint of a blueprint).
  - It has static constants and abstract methods.
- The interface is *a mechanism to achieve abstraction*.
  - There can be only abstract methods in the Java interface, not method bodies.
  - It is used to achieve abstraction and multiple inheritance in Java.
- In other words, you can say that interfaces can have abstract methods and variables but they cannot have a method body.

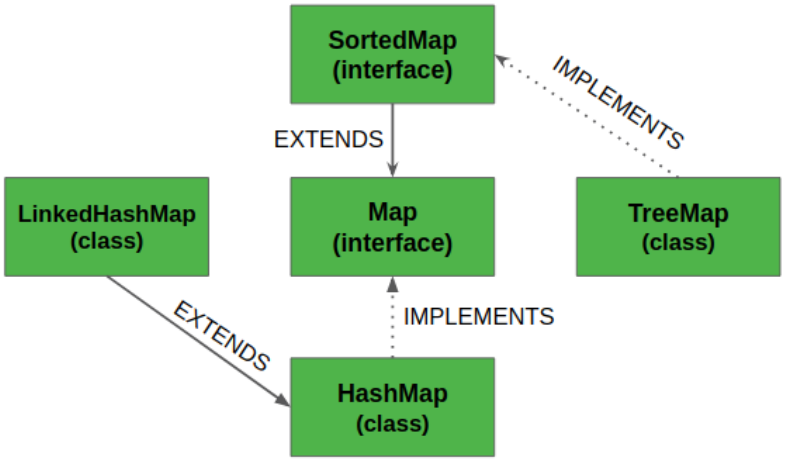
Interfaces (ii)

- Java Interface also **represents the IS-A relationship**.
- Interfaces cannot be instantiated similar to the abstract classes.
  - Since Java 8, **default and static methods can be part** of an interface.
  - Since Java 9, **private methods can be** in an interface.



Usage of interfaces

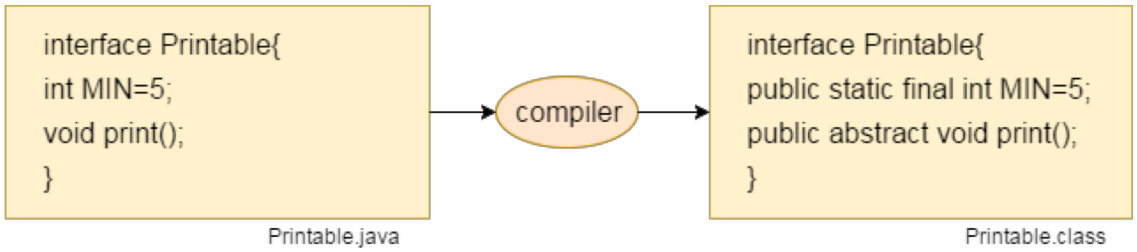
- There are mainly three reasons why to use interfaces.
- 1) To achieve abstraction.
- 2) Interfaces support the functionality of multiple inheritance.
- 3) Can be used to achieve loose coupling.



MAP Hierarchy in Java

How to declare an interface

- An interface is declared by using the interface keyword.
- It provides total abstraction; meaning all the methods in an interface are declared with the empty body, and all the fields are public, static and final by default.
- A class that implements an interface must implement all the methods declared in the interface.
- The Java compiler adds public and abstract keywords before the interface methods. Moreover, it adds public, static and final keywords before data members.

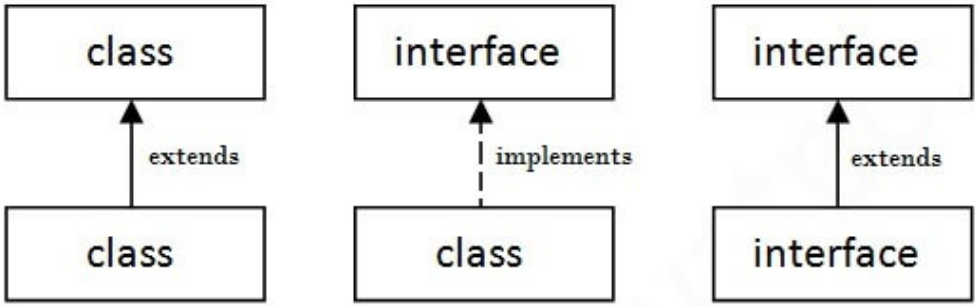



```
interface <interface_name>{

    // declare constant fields
    // declare methods that abstract
    // by default.
}
```

Relationship between classes and interfaces

- A class extends another class, an interface extends another interface, but a **class implements an interface**.



- Example: The Printable interface has only one method, and its implementation is provided in the Aclass class (Aclass)

Interface examples

- In this example, the Drawable interface has only one method. Its implementation is provided by Rectangle and Circle classes.
- In a real scenario, an interface is usually defined by someone else, but its implementation is provided by different implementation providers.
- Moreover, it is used by someone else. The implementation part is hidden by the user who uses the interface (TestInterface1).
- **Bank example** with interfaces (TestInterface2).

Multiple inheritance using interfaces

- If a class implements multiple interfaces, or an interface extends multiple interfaces, it is known as multiple inheritance (Aclass2).

Multiple Inheritance in Java

- Q) Multiple inheritance is not supported through class in java, but it is possible by an interface, why?
  - As we have explained in the inheritance part, multiple inheritance is not supported in the case of a class because of **ambiguity**. However, it is supported in case of an interface because there is no ambiguity. because its implementation is provided by the implementation class (TestInterface3).

Interface inheritance

- An interface can extend another interface to achieve inheritance (Aclass2).
- Default Method in Interface
  - Since Java 8, we can have method bodies in interfaces, but we need to make them default methods.
- Static Method in Interface
  - We can also have static methods in interfaces.

Difference between abstract class and interface

Abstract class	Interface
1) Abstract class can **have abstract and non-abstract** methods.	Interface can have **only abstract** methods. Since Java 8, it can have **default and static methods** also.
2) Abstract class **doesn't support multiple inheritance**.	Interface **supports multiple inheritance**.
3) Abstract class **can have final, non-final, static and non-static variables**.	Interface has **only static and final variables**.
4) Abstract class **can provide the implementation of interface**.	Interface **can't provide the implementation of abstract class**.
5) The **abstract keyword** is used to declare abstract class.	The **interface keyword** is used to declare interface.
6) An **abstract class**can extend another Java class and implement multiple Java interfaces.	An **interface** can extend another Java interface only.
7) An **abstract class**can be extended using keyword "extends".	An **interface class**can be implemented using keyword "implements".
8) A Java**abstract class**can have class members like private, protected, etc.	Members of a Java interface are public by default.

9)**Example:**
public abstract class Shape{
public abstract void draw();
}
	Example:
	public interface Drawable{
	void draw();
	}

- **Let's see an example (differenceTest)**

Where are we now?

- Constructors - Exam
- Static key word - Exam
- This key word - Exam
- Inheritance - Exam
- Aggregation - Exam
- Method overloading and overriding - Exam
- Polymorphism - Exam
- Abstract classes, Interfaces - Exam
- Encapsulation, exception handling
- …Object class, cloning, java math…

God påske!