**National University of Computer & Emerging Sciences, Karachi**

**Computer Science Department**

**Spring 2022, Lab Manual – 08**

|  |  |
| --- | --- |
| **Course Code: CL-217** | **Course : Object Oriented Programming Lab** |
| **Instructor(s) :** | **Abeer Gauher, Hajra Ahmed, Syed Zain ul Hassan** |

**LAB - 8**

# Abstract Classes & Interface

**Abstract Class:**

A class which is declared as abstract is known as an **abstract class**. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

* 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 also.
* It can have final methods which will force the subclass not to change the body of the method.

**Syntax**

***abstract*** *class MyClass  
{ // code  
}*

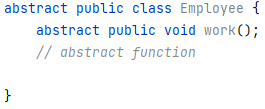
**Abstract Method:**

A method which is declared as abstract and does not have implementation is known as an **abstract method**.

**Syntax**

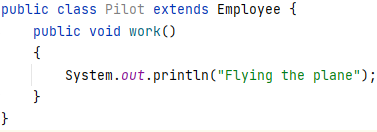
***abstract*** *void myFunction( ); //no method body and abstract*

**Example**

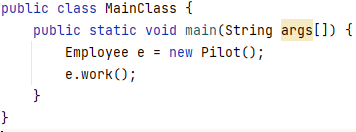


It is the responsibility of child class(es) to override the abstract function and “complete” the parent class.

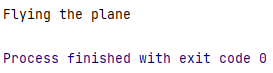
**Example**



**Example: (***Using abstract class from main)*

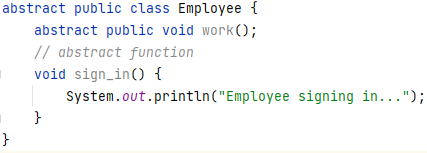


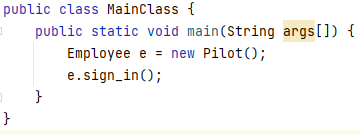
**Output:**



**Point to Remember:** We cannot create an instance of the abstract class.  
**Point to Remember:** An abstract class can contain concrete functions.

**Example:**



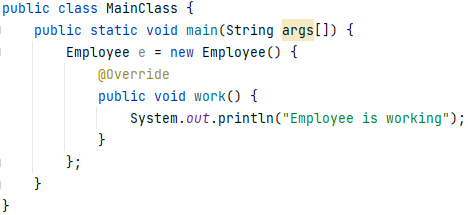


**Anonymous Class:**

If we do not have a child class, we can still override the abstract function by creating an **Anonymous Class.**

* A new class is defined (without a name, so called anonymous class)
* This new class extends abstract base class
* Abstract methods are overriden in this new class
* New instance of this new class is created and assigned to the parent variable

**Example**



**Interfaces:**

An **interface** in Java is a blueprint of a class. It has static constants and abstract methods. A difference between abstract classes and interface is that there can be concrete methods in abstract classes whereas interface cannot contain any.

**How to create an interface:**

An interface is declared by using the interface keyword. It provides total abstraction; means 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.

**Syntax:**

***interface*** *<interface\_name>{*

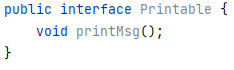
*// declare constant fields*

*// declare methods that are abstract*

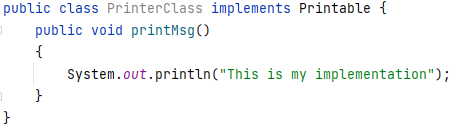
*// by default.*

*}*

**Example:** *(Declaring an interface)*

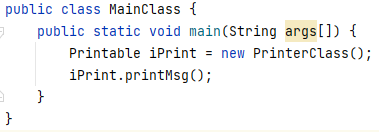


**Example:** *(Implementing interface in class)*



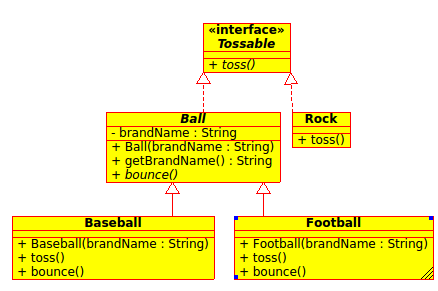
**Point to Remember:** We can create static methods in interfaces. But we must define their implementation inside the interface.  
**Point to Remember:** When a class implements an interface, we can save the object of that class in the interface variable.

**Example:**



**Exercises**

1. Create an abstract class 'Parent' with a method 'message'. It has two subclasses each having a method with the same name 'message' that prints "This is first subclass" and "This is second subclass" respectively. Call the methods 'message' by creating an object for each subclass.
2. Create an abstract class 'Bank' with an abstract method 'getBalance'. $100, $150 and $200 are deposited in banks A, B and C respectively. 'BankA', 'BankB' and 'BankC' are subclasses of class 'Bank', each having a method named 'getBalance'. Call this method by creating an object of each of the three classes.
3. We have to calculate the percentage of marks obtained in three subjects (each out of 100) by student A and in four subjects (each out of 100) by student B. Create an abstract class 'Marks' with an abstract method 'getPercentage'. It is inherited by two other classes 'A' and 'B' each having a method with the same name which returns the percentage of the students. The constructor of student A takes the marks in three subjects as its parameters and the marks in four subjects as its parameters for student B. Create an object for eac of the two classes and print the percentage of marks for both the students.
4. An abstract class has a construtor which prints "This is constructor of abstract class", an abstract method named 'a\_method' and a non-abstract method which prints "This is a normal method of abstract class". A class 'SubClass' inherits the abstract class and has a method named 'a\_method' which prints "This is abstract method". Now create an object of 'SubClass' and call the abstract method and the non-abstract method. (Analyse the result)
5. Create an abstract class 'Animals' with two abstract methods 'cats' and 'dogs'. Now create a class 'Cats' with a method 'cats' which prints "Cats meow" and a class 'Dogs' with a method 'dogs' which prints "Dogs bark", both inheriting the class 'Animals'. Now create an object for each of the subclasses and call their respective methods.
6. Implement the following class hierarchy on paper. You do not need to fill in the method bodies for the *toss* or *bounce* methods.



1. A queue is an abstract data type for adding and removing elements. The first element added to a queue ist the first element that is removed (first-in-first-out, FIFO). Queues can be used, for instance, to manage processes of an operating system: the first process added to the waiting queue is reactivated prior to all other processes (with the same priority).

Design an interface Queue, with methods to add and remove elements (integers). Furthermore, a methode to check whether the queue is empty or not should exist.

* Good luck!