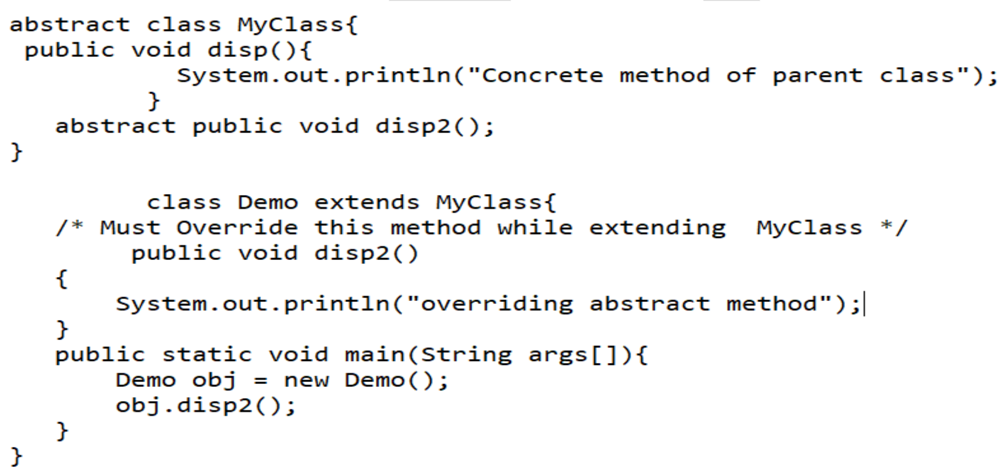
EXPERIMENT NO. 5

1. What is abstract class ? Give an example.

Ans -

An abstract class is a class that is declared abstract it may or may not include abstract methods. In Java, an instance of an abstract class cannot be created , we can have references of abstract class type though. an abstract class can contain constructors and a constructor of abstract class is called when an instance of a inherited class is created.
If class contain at least single abstraction method then class must be declared as an abstract class. An abstract class is mostly used to provide a base for subclasses to extend and implement the abstract methods and override or use the implemented methods in abstract class.

Example –



```
abstract class MyClass{
 public void disp(){
           System.out.println("Concrete method of parent class");
         }
   abstract public void disp2();
}

         class Demo extends MyClass{
   /* Must Override this method while extending  MyClass */
        public void disp2()
   {
       System.out.println("overriding abstract method");
   }
   public static void main(String args[]){
       Demo obj = new Demo();
       obj.disp2();
   }
}
```

In above example disp2() method is declared as an abstract method in MyClass . As we have declared it as an abstract then we have to must implement it . The disp2() method have implementation in demo class.

2.Why to declare class as abstract ?

Ans –

Abstract classes improve the situation by preventing a developer from instantiating the base class , because a developer has marked it as having missing functionality. It also provides compile-time safety so that you can ensure that any classes that extend your abstract class provide the bare minimum functionality to work, and you don't need to worry about putting stub methods that inheritors somehow have to magically know that they have to override a method in order to make it work.

3. What is interface ? Give an example.

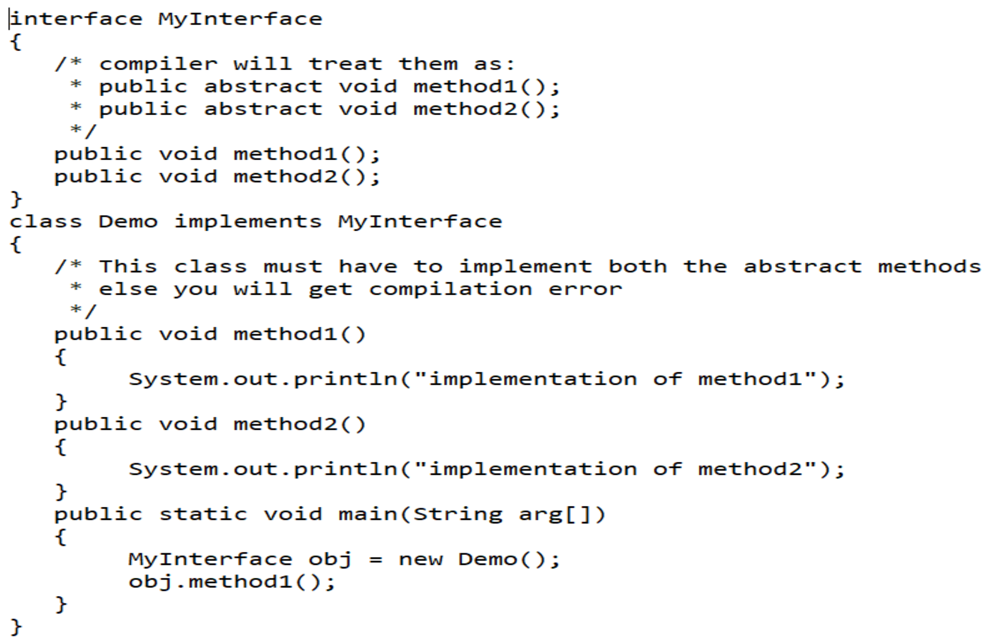
Ans –

An interface is just like Java Class, but it only has static constants and abstract method. Java uses Interface to implement multiple inheritance. A Java class can implement multiple Java Interfaces. All methods in an interface are implicitly public and abstract.
In an interface, you can't instantiate variable and create an object . The interface cannot contain concrete(with implementation) methods .Use an abstract class when a template needs to be defined for agroup of subclasses .Use an interface when a role needs to be defined for other classes, regardless of the inheritance tree of classes

- Syntax for Declaring Interface –
-

```
interface interface_name
{
        //methods
}
```

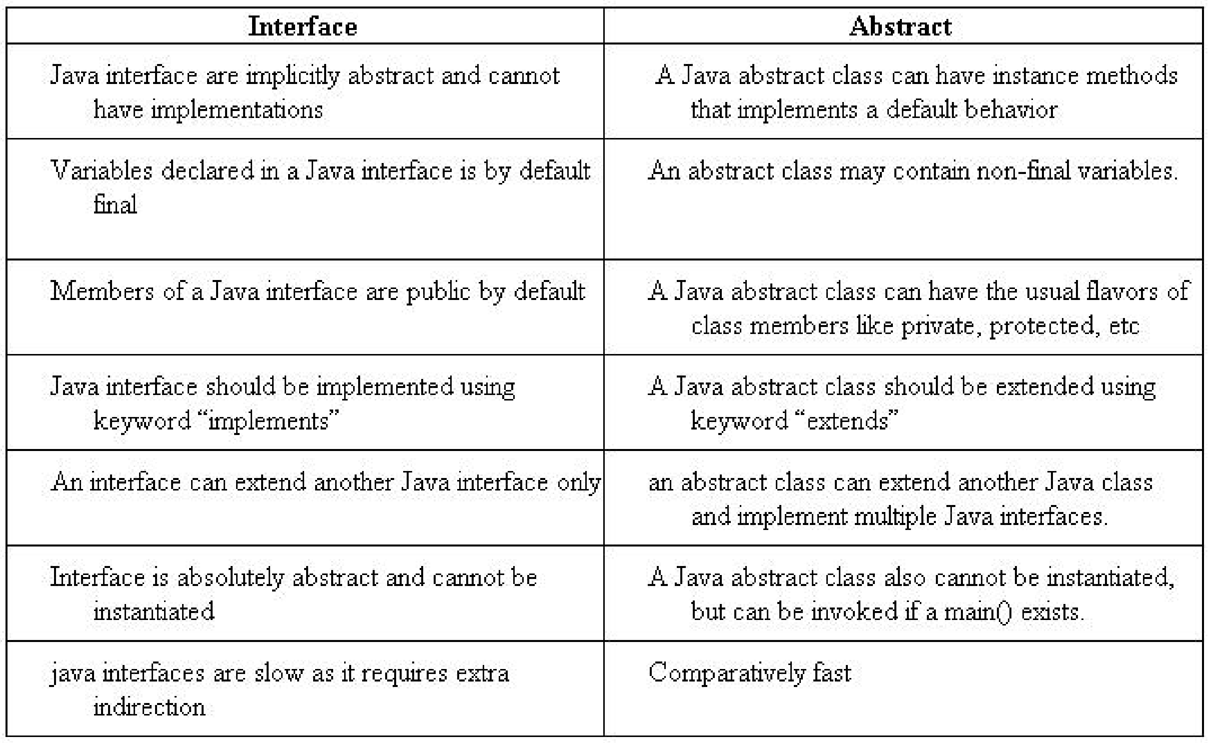
Example –
This is how a class implements an interface. It has to provide the body of all the methods that are declared in interface or in other words you can say that class has to implement all the methods of interface.



```
interface MyInterface
{
   /* compiler will treat them as:
    * public abstract void method1();
    * public abstract void method2();
    */
   public void method1();
   public void method2();
}
class Demo implements MyInterface
{
   /* This class must have to implement both the abstract methods
    * else you will get compilation error
    */
   public void method1()
   {
        System.out.println("implementation of method1");
   }
   public void method2()
   {
        System.out.println("implementation of method2");
   }
   public static void main(String arg[])
   {
        MyInterface obj = new Demo();
        obj.method1();
   }
}
```

4.Differentiate abstract class and interface.

Ans -



Interface	Abstract
Java interface are implicitly abstract and cannot have implementations	A Java abstract class can have instance methods that implements a default behavior
Variables declared in a Java interface is by default final	An abstract class may contain non-final variables.
Members of a Java interface are public by default	A Java abstract class can have the usual flavors of class members like private, protected, etc
Java interface should be implemented using keyword "implements"	A Java abstract class should be extended using keyword "extends"
An interface can extend another Java interface only	an abstract class can extend another Java class and implement multiple Java interfaces.
Interface is absolutely abstract and cannot be instantiated	A Java abstract class also cannot be instantiated, but can be invoked if a main() exists.
java interfaces are slow as it requires extra indirection	Comparatively fast

5 . Why to use interface ?

Ans –

- It is used to achieve total abstraction.
- Since java does not support multiple inheritance in case of class, but by using interface it can achieve multiple inheritance .
- It is also used to achieve loose coupling.
- Interfaces are used to implement abstraction.

6. What is default method and static method inside interface ?

Ans –

1.Default method-

Reengineering an existing JDK framework is always very complex. Modify one interface in JDK framework breaks all classes that extends the interface which means that adding any new method could break millions of lines of code. Therefore, default methods have introduced as a mechanism to extending interfaces in a backward compatible way .Default methods can be provided to an interface without affecting implementing classes as it includes an implementation. If each added method in aninterface defined with implementation then no implementing class is affected. An implementing class can override the default implementation provided by the interface.

2.Static method -

1. Java interface static method is part of interface, we can't use it for implementation class objects.
2. Java interface static methods are good for providing utility methods, for example null check, collection sorting etc.
3. Java interface static method helps us in providing security by not allowing implementation classes to override them.
4. We can't define interface static method for Object class methods, we will get compiler error as "This static method cannot hide the instance method from Object". This is because it's not allowed in java, since Object is the base class for all the classes and we can't have one class level static method and another instance method with same signature.
5. We can use java interface static methods to remove utility classes such as Collections and move all of it's static methods to the corresponding interface, that would be easy to find and use.

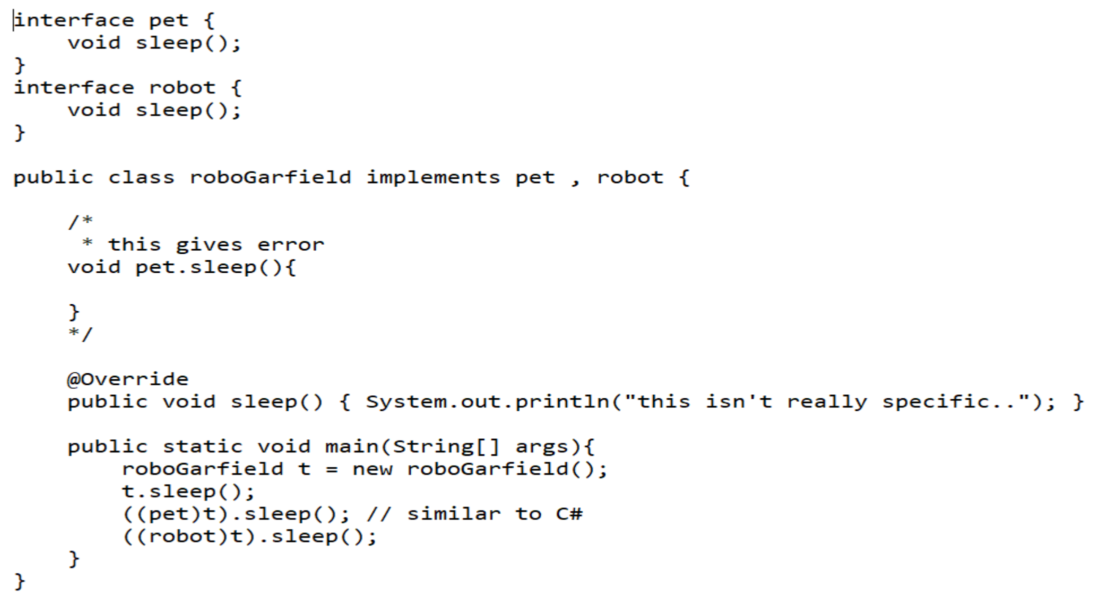
7 . What is explicit interface implementation ?

Ans –

If a class implements two interfaces that contain a member with the same signature, then implementing that member on the class will cause both interfaces to use that member as their implementation. If the two interface members do not perform the same function, however, this can lead to an incorrect implementation of one or both of the interfaces.

It is possible to implement an interface member explicitly - creating a class member that is only called through the interface, and is specific to that interface. This is accomplished by naming the class member with the name of the interface and a period.

Explicit implementation is also used to resolve cases where two interfaces each declare different members of the same name such as a property and a method.



```
interface pet {
    void sleep();
}
interface robot {
    void sleep();
}

public class roboGarfield implements pet , robot {

    /*
     * this gives error
    void pet.sleep(){

    }
    */

    @Override
    public void sleep() { System.out.println("this isn't really specific.."); }

    public static void main(String[] args){
        roboGarfield t = new roboGarfield();
        t.sleep();
        ((pet)t).sleep(); // similar to C#
        ((robot)t).sleep();
    }
}
```

But even though we can cast the roboGarfeild object to its pet or robot type , we cant do an explicit implementation like c#.

Note - java doesn't support explicit interface implementation like C#. But for cases where they can't be avoided.

8.What is marker interface ? What is use of marker interface ?

Ans –

Marker interface in Java is interfaces with no field or methods or in simple word empty interface in java is called marker interface. Example of market interface is Serializable, Clonnable and Remote interface. Marker interface are also called tag interface in Java.

Use of marker interface –

1) Looking carefully on marker interface in Java e.g. Serializable, Clonnable and Remote it looks they are used to indicate something to compiler or JVM. So if JVM sees a Class is Serializable it done some special operation on it, similar way if JVM sees one Class is implement Clonnable it performs some operation to support cloning. Same is true for RMI and Remote interface. So in short Marker interface indicate, signal or a command to Compiler or JVM.

2)Apart from using built in marker interface for making a class Serializable or Clonnable. One can also develop his own marker interface. Marker interface is a good way to classify code. You can create marker interface to logically divide your code and if you have your own tool than you can perform some pre-processing operation on those classes. Particularly useful for developing API and framework like Spring or Struts.After introduction of Annotation on Java5, Annotation is better choice than marker interface and JUnit is a perfect example of using Annotation e.g. @Test for specifying a Test Class. Same can also be achieved by using Test marker interface.

3)One more use of marker interface in Java can be commenting. a marker interface called Thread Safe can be used to communicate other developers that classes implementing this marker interface gives thread-safe guarantee and any modification should not violate that. Marker interface can also help code coverage or code review tool to find bugs based on specified behavior of marker interfaces.Again Annotations are better choice @ThreadSafe looks lot better than implementing ThraedSafe marker interface.