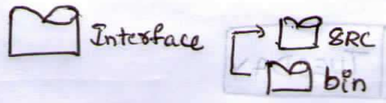
INTERFACE



P71

1)

```
interface Sample1
{
  void test1();
}
class A implements Sample1
{
   public void test1()
   {
     Sop("test1() implemented in Class A");
   }
 }

class Run1
{
  psvm(String[] args)
  {
   Sop("Program starts...");
   A a1= new A();
   a1.test1();
   Sop("Program ends...");
  }
}
```

O/p:

```
Program starts...
test1() implemented in Class A
Program ends...
```

1*) An interface is one of the java type definition block which is 100% abstract.

2*) All interface methods are by default abstract and public

3*) An interface method need not to be declared as abstract bec by default all methods are abstract.

4*) We cannot develop static methods inside interface.

5*) An interface methods cannot be declared as final.

6*) Interface variable has to be initialized at the time of declaration.

7*) By default the interface itself is a abstract.

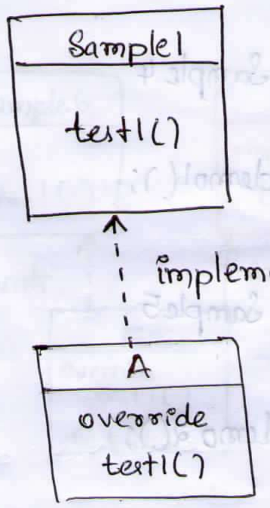
8*) Compiler generates Class file for interface definition block.

9*) An interface methods can get body or implementations in Sub classes.

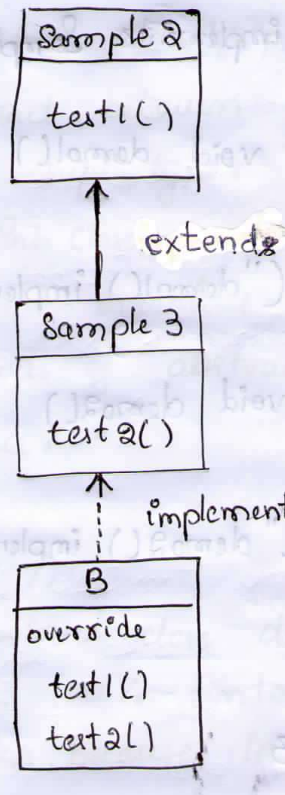
2) P72

```
interface Sample2
{
  void test1();
}
interface Sample3 extends Sample2
{
  void test2();
}
Class B implements Sample3
{
  public void test1()
  {
    Sop("test1() implemented in Class B");
  }
  public void test2()
  {
    Sop("test2() implemented in ClassB");
  }
}
Class Run2
{ psvm (String[] args)
  {
    Sop ("Program starts...");
    B b1 = new B()
    b1.test1();
    b1.test2();
    Sop ("Program ends...");
  }
}
```

Ex 1)

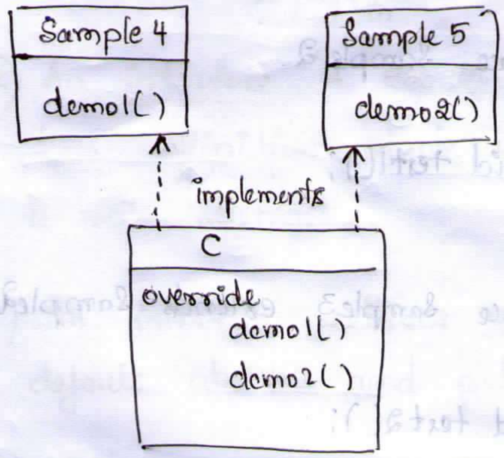


Ex 2)



3) P73

```
interface Sample4
{
    void demo1();
}

interface Sample5
{
    void demo2();
}

Class C implements Sample4, Sample5
{
  public void demo1()
  {
    Sop("demo1() implemented in Class C");
  }
  public void demo2()
  {
    Sop("demo2() implemented in Class C");
  }
}

Class Run3
{
  psvm(String[] args)
  {
    Sop("Program starts...");
    C c1 = new C();
    c1.demo1();
    c2.demo2();
    Sop("Program ends...");
  }
}
```



10*) Sub class inheriting an interface should implement all the abstract methods of interface, otherwise sub class becomes abstract.

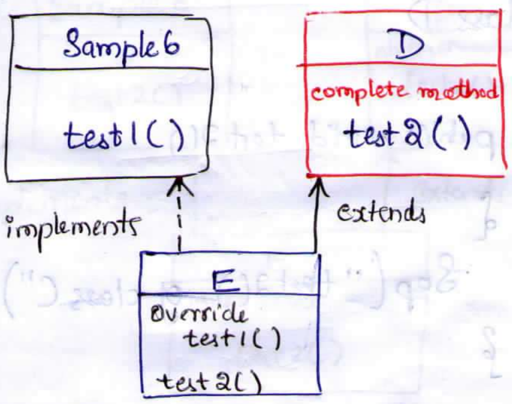
11*) A subclass can inherite more then one interface, in such case the sub-class should overwrite all the interface methods.

12*) A class can inherit from both class and interface.

If the super class is abstract then sub-class should override both abstract class and interface methods().

4) P74

```
Interface Sample6
{
  void test1();
}

class D
{
   void test2()
   {
     Sop("running test2() method");
   }
}

Class E extends D implements Sample6
{
   public void test1()
   {
     Sop("~~running~~ test1() implemented in ClassE");
   }
   void test2()
   {
     Sop ("running test2() in Class E");
   }
}
class Run4
{
  psvm (String[] args)
  {
    Sop ("Program starts...");
    E e1 = new E();
    e1. test1();
    e1 . test2();
    Sop ("Program ends...");
  }
}
```



13*) An abstract class as well as interface describes/defines the contract between the sub class and super type. The contract is Sub Class can exsist only if it overrides the abstract methods of abstract class or Interface.

14*) Abstract class doesn't enforce 100% contract with sub-class because in abstract class we can develop concrete methods.

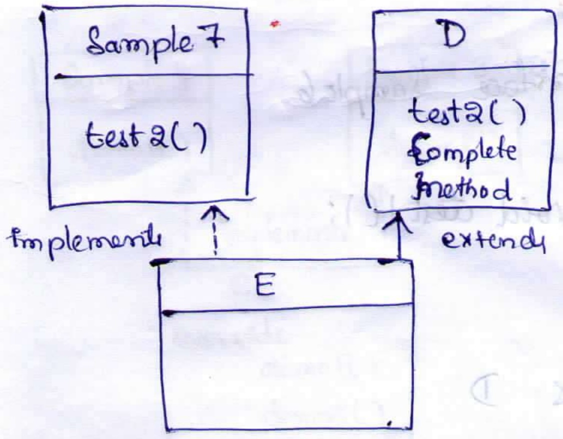
15*) Interface enforces 100% contract with sub classes because inside interface we cannot develop Concrete Methods().

O/p:
```
Program starts...
test1() implemented in Class E
running test2() in Class E
Program ends...
```

P75

5)
```
class D
{
  public void test2()
  {
    Sop("test2() of class C");
  }
}

interface Sample7
{
   void test2();
}

Class E extend D implents Sample 7
{

}

Class Run4
{
  public vsm ( )
  {
    Sop("Program starts...");
    E e1 = new E();
    e1. test2();
    Sop ("Program ends...");
  }
}
```

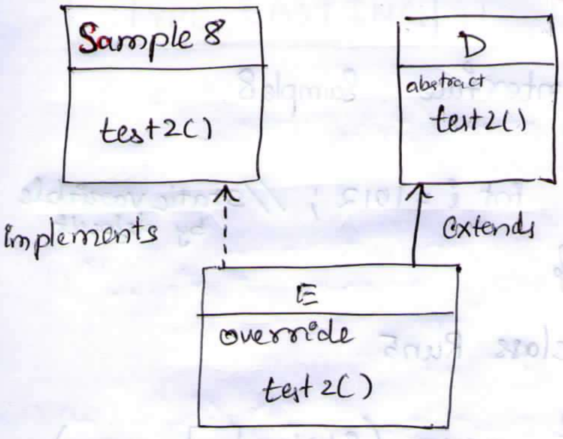
O/p:
```
Program starts...
test2() of class C
Program ends...
```



16*) While developing an application if sub class behavior changes, then those behaviors are defined as abstract in interface, so that the sub class can implement according to sub-class specification.

P76

6) abstract class D
```
{
  abstract void test2();
}

interface Sample8
{
    void test2();
}

class E  extends D implements Sample6
{
    public void test2()
    {
        Sop(" test2() implemented in class E");
    }
}

Class Run4
{
    psvm( )
    {
        Sop("Program starts...");
        E e1 = new E();
        e1. test2() ;
        Sop ("Program ends...");
    }
}
```



O/p :
```
Program starts...
test2() implemented in class E
Program ends...
```

P77

7)
```
interface Sample8
{
    int i = 1012; //static variable
                    by default
}
class Run5
{
    psvm (String[] args)
    {
      Sop("Program starts...");
      Sop("i=" + Sample8.i);
      Sample8.i = 1227; //error;
     // interface variable are final
     // can't reassign.
      Sop("i=" + Sample8.i);
      Sop ("Program ends...");
    }
}
```

17*) All interface variable by default final and static

****

1. * Java understands only Homogenous

LHS = RHS

int i = 10;
double d = 21.66;
boolean b = true;

1) int i;
type ← read

how to read? i is of integer type

2) double d; d is of double type
type

3) boolean b; b is of boolean type

2* hetrogenous assignment statements

LHS ≠ RHS

int K = 21.66;
double b = 10;