Name - Sakshi Sharma
Course - M.Sc. Computer Science
Rou No. - 53
Subject - Java (Minor)
Date - 18-June-2021

Ans. (1)

```
1. public class Temperature {
2.     public static void main(String args[]){
3.         double fahrenheit = 62.5;
4.         /* Convert */
5.         double celsius = f2c(fahrenheit);
6.         System.out.println(fahrenheit + 'F' + "=" +
                                celsius + 'C');
7.     }
8.     Static double f2c(double fahr){
9.         return (fahr - 32) * 5/9;
10.    }
11. }
```

line1 :- Java is a case sensitive language. Keywords public and class should be in lowercase only. And the file name is supposed to be same as the class in which main function is defined. Since, our filename is Temperature.java. Therefore, class name should be Temperature only.

line2 :- public is a keyword which is supposed to be in lowercase only.
main should be declared as static only because JVM call main() before any object of the class are made. And to call a function of a class without creating an object of the class. function should be declared as static.
String is a keyword for defining a variable of String type. It cannot be written as string.

main function parameter should be an array of String not a String. Therefore, [] is must.

Line 4 :- Proper format of writing a comment is -
/* Convert */

Line 6 :- Celsius is not defined it should celsius where 'c' is ~~in~~ ~~small~~ lowercase letter.

Line 8 :- lossy conversion from double to float. Java does not allow this. It should be double only. static function main cannot call non-static fn fac.

Line 9 :- return is a keyword. It cannot be written as RETURN.

Ans (2) Function Overloading - When we have a more than one function with same name but they differ in only types and/or number of parameters. Then, we say the function is overloaded. Their return type may or may not be same.

Function Overriding - When a same function in super class ~~is overloaded in~~ and subclass both. Then we say that function is overridden. The name and parameters of function are same in both sub class and superclass.

Example :-
```
class A {
        void show()
        {
        }
}
Class B extends A {
```

```
    void show()  // Overridden
    {

    }
    void show(String msg) // Overloaded.
    {
    }
}
```

In this example, show() is overridden because super class A and subclass B both have same name and signature of show(). But show(String msg) in class B is not overridden because there is not same function with same signature in Class A.

function show(String msg) is overloaded in class B. because class B has two show() one without parameter and other with one String parameter.

Ans. (3) Advantages of Abstract class over interfaces. —

* Abstract class can have abstract & non-abstract methods. but interfaces can have only abstract methods.
* Abstract class can have final, static, non-final and non-static variables but interfaces can have only static and final variables.
* Abstract classes can provide the implementation for interfaces but interfaces cannot provide the implementation for abstract class.
* Abstract class members can have access modifiers as private, protected, public etc. but members of interfaces are public by default.

* Abstract classes can extend another class and implement multiple interfaces but interface can only extend another java interface.

Advantages of interfaces over abstract classes.

* Interface support multiple inheritance but abstract classes does not.

*

Ans (5) OOPs supports abstraction, Encapsulation, inheritance and polymorphism.

Abstraction - It is a process of hiding implementation details and representing only essential features.

Encapsulation - The wrapping of code and data together into one single unit is known as encapsulation.

Inheritance - It is a process by which one object acquires the properties and behaviours of another objects.

Polymorphism - It is a process of definining a function for more than one purposes.

A language that supports all the four features of OOP is known as Object Oriented Programming language, Example- Java where every task is performed inside a class and we make objects of class to perform the tasks.

* Java achieves abstraction by hiding private data members.

* Java achieves encapsulation by wrapping all the methods and variables within a class.

* Java achieves inheritance using 'extends' keyword. One class can inherit another class using extend

keyword. Multiple inheritance is not allowed in java.

* Java achieves polymorphism using function overloading and function overriding.

Ans (6) Uses of ~~static~~ final keyword

* It is equivalent to named constant. If a variable is declared as final then its value cannot be changed. ~~If~~ Value of that variable can only be given at the defination time or inside a constructor.

Eg- final int var = 5;

* It is used to prevent overriding of method in derived class.

Eg - final void show() // in class A.
{
}

It cannot be overriden in class B.

* If final is used with a class. It prevent a class from being inherited.

Eg. final class A {

}

class B extends A {

}

// error

Use of static keywords

* Static members can be accessed using class name or by object references. All objects of a class share the same copy of the static members.
* When a member is declared as static, it can be accessed before any object of its class are created, using class name.
* A static variable/method belongs to the class and not to any of the objects of the class.
* Example -

```
class A {
        static int a=5;
        static int b=10;
        static void show() {
                System.out.println("a= "+a);
        }
}
Class B {
        public static void main (String args[])
        {
                A.show(); // static method
                System.out.println("b= "+A.b); // static var
        }
```

O/P - a=5
b=10

In the above example, show() of class A is called in class B using class name only, we have not created any reference/object of class A.

Static member 'b' is also accessed in class B directly using class name only.

* Java also support static block that gets executed exactly once, when the class is first loaded.
* Static ~~members~~ methods can only call other static methods.

Ans ④ * this keyword is used to call other constructor of same class.

for example, you can call default constructor from a parameterised constructor using this keyword.

Eg - Class A

```
{   int a;
    int b;
    A()
    { a=5;
    { b=5;
    A(int a)
    {   this();
        a=10;
    }
```

* Super keyword is used to call base class constructor from derived class constructor. but it should be the first statement in derived class constructor.

* super keyword can be used to access hidden members of base class in derived class using super keyword. member should not be private in base class.

* this keyword can be used to refer to the class data member.

Eg. class A

```
{ int x;
  A(int x)
  { this.x=x;
  }
```

Example of (i) :- & (ii) :-

```
class A {  int a;
           A(int x)
           {
              a=x;
           }
        }
class B extends A {

        B(int x)
        {
           super(x); // call base class A constructor
           System.out.println(super.a); //a of super class
        }
        public static void main(String args[])
        {
           B ob = new B(5);
        }
     }
```

O/P - 5 // 'a' is accessed in derived class B of super class.

⇒ Eg:

```
class A {
        void show()
        {
           System.out.println("In class A");
        }
     };
class B extends A {
        void show()
        {
           super.show();
           System.out.println("In class B");
        }
        public static void main(String args[])
        {
           B ob = new B();
           ob.show();
        }
```

O/P = In class A
In class B