MCS0201

JAVA PROGRAMMING

Minor Test

Name: IRFAN SHEIKH

Roll no: 16

1.)

```
public class Temperature {
    public static void main (String [] args) {
        double fahrenheit = 62.5;
        /* Convert */
        System.out.println (fahrenheit + "F" + "=" + celsius + "C");
    }

    static double f2c (double fahr) {
        return ( fahr - 32 ) * 5 / 9;
    }
}
```

- ~~Fi Co~~ Class name is case sensitive
- public, class, ~~cel~~ return, should be in lower case
- main method:
  - should be static
  - should have String[] args as parameters.
- f2c should have double in params

2.) Method calling mechanisms

- **Overloading** : when an overloaded method is called, Java uses the type and / or number of arguments to find out which version of the overloaded method is to be called. The return type alone is not sufficient to distinguish two versions of an overloaded method. When a call to an overloaded method is made, the version whose parameters match the arguments in the call is invoked. In the context of objects, the declared type of the reference at compile time is used to determine which method will be executed at runtime.

- **Overriding** : Java uses the concept of Dynamic Method Dispatch to resolve calls to overridden methods at run-time. A superclass reference variable can refer to a subclass object. When an ~~overloaded~~ overridden method is called through a superclass reference, Java examines the type of the object being referred to at the time of the call, and decides which method to execute at runtime.
Therefore, if a subclass overrides a superclass method, then when different types of objects are referred to through a superclass reference variable, different versions of the method are actually called.

3.) Advantages of abstract classes over interface

1.) Abstract classes can have both abstract and non-abstract methods, whereas interfaces can only have abstract methods.

2.) They can have final, non-final, static, and non-static members, whereas interface can have only static and final variables.

3.) They can provide the implementation of an interface, whereas interfaces can't provide the implementation of abstract class.

4.) They can extend another Java class and implement multiple Java interfaces, whereas interfaces can only extend other interfaces.

5.) They can have private, protected, and public members, whereas members of a Java interface are public by default.

Advantages of interfaces over abstract classes:

1.) Interfaces support multiple inheritance, whereas abstract classes don't.

2.) Interfaces can be used to provide common functionality to otherwise unrelated classes.

3.) Java doesn't support multiple inheritance using classes, but we can achieve this effect by implementing multiple interfaces.

4.) • this keyword can be used to refer to the current (invoking) object.

• super keyword can be used to access a member of the superclass, or to call the superclass constructor. super is useful when member names of a subclass hide members by the same name in the superclass.

eg:
```
class A {
    void m1( ) { ... }
}

class B extends A {
    B( ) {
        super( ); // call A's constructor
    }

    void m1( ) { ... }      // overrides A's m1 method
    void mA1( ) { super.m1( ) }
    public static void main (String[ ] args) {
        B b = new B( );
        b.mA1( );   // calls super.m1( ).
    }
}
```

5.) Basic concepts of OOP

(i) Abstraction : a way to manage complexity is through abstractions. eg: ~~we don't know~~ In the context of a vehicle, we don't need to know how the internal systems like engine, brakes, etc, work. All we need to know is how to drive and we can utilize the vehicle as per our needs. We have abstracted away the internal workings.

2.) Encapsulation : binds together code and the data it manipulates, and prevents the code and data from being arbitrarily accessed by outside code. Access to the code and data is tightly controlled through a well defined interface

eg: the only way to control the automatic transmission of a vehicle is through a well defined interface: the gear shift stick.

3.) Inheritance : the process by which one object acquires the properties of another object.

eg: An automatic transmission car is part of the class Car, which in turn is part of the vehicle class.
A car has all the properties of a vehicle.

4.) Polymorphism allows one interface to be used for a general class of actions.

eg: A method to add two integers has the same logic as a method that adds two floats. Only the type of data has changed, the interface (method name) is the same.

6.) • Static keyword is used to define a class member that can be used independently of any object of that class, i.e. without creating any objects. • All instances of this class share the same copy of that member

eg:
```
class A {
    static int id = 0;
    A () {
      id ++;
    }
}
class B {
    public static void main (String[] args){
        A a1 = new A();
        A a2 = new A();
        System.out.println ("Number of objects of class A: "+ A.id);
    }
}
```

The main method is also static, which means it can be accessed as B.main();

- final keyword : can be used to :

(i) define constants. eg: final int num = 10;

(ii) a final reference to always refers to the same object.

```
final A a = new A();
```

(iii) a method declared final can't be overridden.

```
class A {
    final void m1() { . . . }
}
class B extends A {
    // can't override m1
}
```

(iv) a class declared final can't be inherited.

```
final class A {
    . . .
}
class B extends A { . . . }
```

invalid !