4.1 Interfaces

4.1.1 Defining Interface

- Using the keyword interface, you can fully abstract a class' interface from its implementation.
- That is, using interface, you can specify what a class must do, but not how it does it.
- Interfaces are syntactically similar to classes, but they lack instance variables, and their methods are declared without any body. In practice, this means that you can define interfaces that don't make assumptions about how they are implemented.
- Once it is defined, any number of classes can implement an interface. Also, one class can implement any number of interfaces.
- To implement an interface, a class must create the complete set of methods defined by the interface.
- However, each class is free to determine the details of its own implementation.
- By providing the interface keyword, Java allows you to fully utilize the "one interface, multiple methods" aspect of polymorphism.

- **Syntax** for defining interface is:

```
access interface InterfaceName
{
      return_type method_name1(parameter list);
      return_type method_name2(parameter list);
      type final-variable1 = value1;
      type final-variable2 = value2;
```

```
        :
        :
        return_type method_namen(parameter list);
        type final-variable n = value n;
}
```

Example:

```
interface area
{
        static final float pi=20;
        void display();
}
```

- ❒ Interfaces have the following properties:
  - ➢ An interface is implicitly abstract. You do not need to use the abstract keyword while declaring an interface.
  - ➢ Each method in an interface is also implicitly abstract, so the abstract keyword is not needed.