Introduction to JAVA Programming

Rathindra Nath Dutta

Senior Research Fellow
Advanced Computing & Microelectronics Unit
Indian Statistical Institute, Kolkata

October 8, 2020

Outline

1. Inheritance (cntd.)
   - Abstract Base Class
   - `final` keyword
   - `Object` class

Abstract Base Class

- Sometimes we want to define a (base) class that declares the structure of a given abstraction without providing a complete implementation of every method (generally unable to)
- This leaves it to each derived class to fill in the details of the unimplemented method(s)

Abstract Base Class

- Sometimes we want to define a (base) class that declares the structure of a given abstraction without providing a complete implementation of every method (generally unable to)
- This leaves it to each derived class to fill in the details of the unimplemented method(s)
- This is achieved by declaring a class as `abstract` and placing one or more abstract methods in it

```
public abstract class Shape {
        public abstract void area();
}
```

Abstract Base Class

- Sometimes we want to define a (base) class that declares the structure of a given abstraction without providing a complete implementation of every method (generally unable to)
- This leaves it to each derived class to fill in the details of the unimplemented method(s)
- This is achieved by declaring a class as `abstract` and placing one or more abstract methods in it

```
public abstract class Shape {
        public abstract void area();
}
```

- It is a restricted class that cannot be instantiated

```
Shape obj; //OK
obj = new Shape(); //Error
```

Abstract Base Class

- An abstract base class (ABC) may have member variables and other non-abstract methods
- ABCs are used to enforce abstraction
- Placing an abstract method forces any concrete (non-abstract) class extending the abstract class to must provide definitions for all of the unimplemented abstract methods in the ABC

```
public class Square extends Shape {
        private int side;
        public Square(int side) {this.side = side;}

        @Override
        public void area() {System.out.println(side*side);}
}
```

Abstract Base Class

- An abstract base class (ABC) may have member variables and other non-abstract methods
- ABCs are used to enforce abstraction
- Placing an abstract method forces any concrete (non-abstract) class extending the abstract class to must provide definitions for all of the unimplemented abstract methods in the ABC

```
public class Square extends Shape {
        private int side;
        public Square(int side) {this.side = side;}

        @Override
        public void area() {System.out.println(side*side);}
}

        Shape obj = new Square(); //allowed
```

Access Specifier: `final`

- A field can be declared as final to prevent its content from being modified; essentially makes it a constant
- Making a method final prevents it from being overridden in some derived class
- Such final methods can enhance performance: compiler is free to make inline calls to them, early binding is possible
- Declaring a class as final prevents it from being inherited; it implicitly declares all of its methods as final too
- It is illegal to declare a class as both abstract and final

Object class

- All Java classes are subclasses of `Object` class
- `Object` class has several methods, few are:

```
public Object()                    String toString()
protected Object clone()           int hashCode()
boolean equals(Object obj)         Class<?> getClass()
protected void finalize()          void wait()
void notify()                      void wait(long timeout)
void notifyAll()                   void wait(long timeout, int r
```

- For more details click here or manually visit the official API documentation page