Object Oriented Programming

Packages and Interfaces

Packages

- Packages are containers for classes
- They are used to keep the class name space compartmentalized. For example, a package allows you to create a class named List, which you can store in your own package without concern that it will collide with some other class named List stored elsewhere
- Packages are stored in a hierarchical manner and are explicitly imported into new class definitions

Packages

- Java provides a mechanism for partitioning the class name space into more manageable chunks. This mechanism is the package
- The package is both a naming and a visibility control mechanism
- You can define classes inside a package that are not accessible by code outside that package

Packages

- To create a package is quite easy: simply include a package command as the first statement in a Java source file
- This is the general form of the package statement:

```
package pkg;
```

- Here, pkg is the name of the package.

Packages

e IDE
/indow Help

Light.java | Scannerint.java | pwr.java | bank.java | charge.java | circle.java | p.java | student.java | accountbalan...

```java
package mypkg;
class balance{
    String name;
    double bal;
    balance(String n, double b){
        name=n;
        bal=b;
    }
    void show() {
        if(bal<0) {
            System.out.println("negative balance");
        }
        else {
            System.out.println(name+" $ " +bal);
        }
    }
}
public class accountbalance {

    public static void main(String[] args) {
        balance current[]= new balance[3];
        current[0]=new balance("Rafsun", 123.33);
        current[1]=new balance("Breanne", 150.55);
        current[2]=new balance("Chirs", 180.55);
        for(int i=0; i<3; i++) {
            current[i].show();
        }// TODO Auto-generated method stub

    }

}
```

Problems | Javadoc | Declaration | Console

```
<terminated> accountbalance [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 15, 2019, 7:19:01 PM)
Rafsun $ 123.33
Breanne $ 150.55
Chirs $ 180.55
```

Protected member example

Vindow Help

testiface.java | stack.java | queue.java | shape.java | NestedIFDem... | defaultmeth... | primenumber....

```java
package mypkg;
class A18{
    protected String bookname;
    protected int bookid;
    protected double customerbill;
}
class B18 extends A18{
    void show() {
        System.out.println("The bookname is"+bookname);
        System.out.println("bookid is"+bookid);
        System.out.println("Customer bill is"+customerbill);
    }
}
public class protectdemo {

    public static void main(String[] args) {
        B18 b1= new B18();
        b1.bookname="Angels and demons";
        b1.bookid=1001;
        b1.customerbill=48.00;
        b1.show();// TODO Auto-generated method stub

    }

}
```

Javadoc | Declaration | Console

```
<terminated> protectdemo [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 18, 2019, 12:36:07 AM)
The bookname isAngels and demons
bookid is1001
Customer bill is48.0
```

Interfaces

- 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, as a general rule, their methods are declared without any body

Interfaces

- In practice, 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.

Interfaces

- To implement an interface, a class must provide the complete set of methods required 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.

Defining an interface

- An interface is defined much like a class. This is a simplified general form of an interface:

```
access interface name {
    return-type method-name1(parameter-list);
    return-type method-name2(parameter-list);

    type final-varname1 = value;
    type final-varname2 = value;
    //...
    return-type method-nameN(parameter-list);
    type final-varnameN = value;
}
```

Interface

- Here is an example of an interface definition. It declares a simple interface that contains one method called callback( ) that takes a single integer parameter.

```
interface Callback {
  void callback(int param);
}
```

Interface

- Once an interface has been defined, one or more classes can implement that interface
- To implement an interface, include the implements clause in a class definition, and then create the methods required by the interface

```
class Client implements Callback {
  // Implement Callback's interface
  public void callback(int p) {

    System.out.println("callback called with " + p);
  }
}
```

Accessing Implementations Through Interface References

- You can declare variables as object references that use an interface rather than a class type
- Any instance of any class that implements the declared interface can be referred to by such a variable
- When you call a method through one of these references, the correct version will be called based on the actual instance of the interface being referred to

Accessing Implementations Through Interface References

- The following example calls the callback( ) method via an interface reference variable:

```
class TestIface {
  public static void main(String args[]) {
    Callback c = new Client();
    c.callback(42);
  }
}
```

Interface example

Window Help

Light.java | Scannerint.java | pwr.java | bank.java | charge.java | circle.java | p.java | accountba

```java
package mypkg;
interface callback{
    void callback(int p);
}
class client implements callback{
    public void callback(int p) {
        System.out.println("Callback called with "+p);
    }
    void nonifacemeth(int z) {

        System.out.println(z);
    }

}
class anotherclient implements callback{
    public void callback(int p) {
        System.out.println("Another version of callback");
        System.out.println("P squared is "+(p*p));
    }
}
public class testiface {

    public static void main(String[] args) {
        callback c = new client();// TODO Auto-generated method stub
        anotherclient ob=new anotherclient();
        c.callback(42);
        c=ob;
        c.callback(42);
    }

}
```

Problems | Javadoc | Declaration | Console

```
<terminated> testiface [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 15, 2019, 7:55:10 PM)
Callback called with 42
Another version of callback
P squared is 1764
```

Interface example

'indow Help

Light.java | Scannerint.java | pwr.java | bank.java | charge.java | circle.java

```java
package mypkg;
interface intstack{
    void push(int item);
    int pop();
}
class fixedstack implements intstack{
    private int stck[];
    private int tos;
    fixedstack(int size){
        stck=new int[size];
        tos=-1;
    }
    public void push(int item) {
        if(tos==stck.length-1)
            System.out.println("Stack is full");
        else
            stck[++tos]=item;
    }
    public int pop() {
        if(tos<0) {
            System.out.println("Stack underflow");
            return 0;
        }
        else
            return stck[tos--];
    }
}
```

Interface example

```java
public class stack {

    public static void main(String[] args) {
        intstack stack1= new fixedstack(5);
        intstack stack2=new fixedstack(8);
        for(int i=0; i<5; i++) {
            stack1.push(i);
        }
        for(int i=0; i<8; i++) {
            stack2.push(i);
        }
        System.out.println("Stack in mystack1");
        for(int i=0; i<5;i++) {
            System.out.println(stack1.pop());
        }
        System.out.println("Stack in mystack2:");// TODO Auto-generated method stub
        for(int i=0; i<8;i++) {
            System.out.println(stack2.pop());
        }
    }

}
```

Problems @ Javadoc Declaration Console

```
<terminated> stack [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 15, 2019, 8:12:07 PM)
Stack in mystack1
4
3
2
1
0
Stack in mystack2:
7
6
5
4
3
2
1
0
```

Interface example

indow Help

Light.java | Scannerint.java | pwr.java | bank.java | charge.java | accountbalan...

```java
package mypkg;
interface icharq{
    void put(int ch);
    int get();
}
class fixedqueue implements icharq{
    private int q[];
    private int putloc,getloc;
    fixedqueue(int size){
        q=new int[size];
        putloc=getloc=0;
    }
    public void put(int ch) {
        if(putloc==q.length) {
            System.out.println("Queue is full");
        }
        else {
            q[putloc++]=ch;
        }
    }

public int get() {
    if(getloc==putloc) {
        System.out.println("Queue is empty");
        return 0;
    }
    else {
        return q[getloc++]; }
}
}
```

Interface example

```java
public class queue {

    public static void main(String[] args) {
        icharq q1= new fixedqueue(5);
        icharq q2=new fixedqueue(8);
        for(int i=0; i<5; i++) {
            q1.put(i);
        }
        for(int i=0; i<8; i++) {
            q2.put(i);;
        }
        System.out.println("queue in myqueue1");
        for(int i=0; i<5;i++) {
            System.out.println(q1.get());
        }
        System.out.println("queue in myqueue2:");// TODO Auto-generated method stub
        for(int i=0; i<8;i++) {
            System.out.println(q2.get());
        }// TODO Auto-generated method stub

    }

}
```

Problems @ Javadoc Declaration Console

```
<terminated> queue [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 15, 2019, 8:31:40 PM)
queue in myqueue1
0
1
2
3
4
queue in myqueue2:
0
1
2
3
4
5
6
7
```

Interface can be extended

- One interface can inherit another by use of the keyword extends
- The syntax is the same as for inheriting classes. When a class implements an interface that inherits another interface, it must provide implementations for all methods

Example

indow Help

Light.java | Scannerint.java | pwr.java | bank.java | testiface.java | stack.java | queue.java | sort.java | shape.jav

```java
package mypkg;
interface rectangle{
    double area();
}
interface rec extends rectangle{
    double perimeter();

}
interface square extends rec{
    double a();
    double p();
}
class s implements square{
    double length, width, side;
    s(double l, double w, double s){
        this.length=l;
        this.width=w;
        this.side=s;
    }
    public double area() {
        return length*width;
    }
    public double perimeter() {
        return 2*(length+width);
    }
    public double a() {
        return side*side;
    }
    public double p() {
        return 4*side;
    }
}
public class shape {

    public static void main(String[] args) {
        s s1= new s(20,10, 5);
        System.out.println(s1.area());
        System.out.println(s1.perimeter());
        System.out.println(s1.a());
        System.out.println(s1.p());

    }

```

Problems | Javadoc | Declaration | Console

```
<terminated> shape (1) [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 17, 2019, 8:53:42 PM)
200.0
60.0
25.0
20.0
```

Example

IDE

Window Help

Light.java | testiface.java | stack.java | queue.java | sort.java

```java
package mypkg;
interface prime{
    void prime(int n);
}
interface primerange extends prime{
    void primerange(int n1, int n2);
}
```

Window Help

testiface.java | stack.java | queue.java | sort.java | shape.java | NestedIFD

```java
class prim implements primerange{
    public void prime(int n) {
        int flag=1;
        for(int i=2; i<=n/2; i++) {
            if(n%i==0) {
                flag=0;
                break;
            }

        }
        if(flag==1) {
            System.out.println(+n+" is a prime number");
        }
        else {
            System.out.println(+n+" is not a prime number");
        }
    }
    public void primerange(int n1, int n2) {
        for(int i=n1; i<n2;i++) {
            int flag=1;
            for(int j=2; j<=i/2; j++) {
                if(i%j==0) {
                    flag=0;
                    break;
                }

            }
            if(flag==1) {
                System.out.println(i);
            }
        }
    }
}
```

Example

```java
public class primenumber {

    public static void main(String[] args) {
        prim p= new prim();
        p.prime(5);
        p.prime(9);// TODO Auto-generated method stub
        p.primerange(10, 50);
    }

}
```

Javadoc Declaration Console

```
<terminated> primenumber [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 17, 2019, 10:29:29 PM)
5 is a prime number
9 is not a prime number
11
13
17
19
23
29
31
37
41
43
47
```

Nested interfaces

- An interface can be declared a member of a class or another interface. Such an interface is called a member interface or a nested interface
- A nested interface can be declared as public, private, or protected
- This differs from a top-level interface, which must either be declared as public or use the default access level, as previously described

Example

```java
package mypkg;
import java.util.Scanner;
class A{
    public interface NestedIF{
        boolean isNotNegative(int x);
    }
}
class B implements A.NestedIF{
    public boolean isNotNegative(int x) {
        return x<0?false:true;
    }
}
public class NestedIFDemo {

    public static void main(String[] args) {
        int t1;
        A.NestedIF nif=new B();// TODO Auto-generated method stub
        Scanner input=new Scanner(System.in);
        t1=input.nextInt();
        if(nif.isNotNegative(t1)) {
            System.out.println("The number is not negative");
        }
        else {
            System.out.println("The number is negative");
        }
    }

}
```

```
<terminated> NestedIFDemo [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 17, 2019, 9:08:44 PM)
-5
The number is negative
```

Default interface method

- As explained earlier, prior to JDK 8, an interface could not define any implementation whatsoever
- The release of JDK 8 changed this by adding a new capability to interface called the default method
- A default method lets you define a default implementation for an interface method

Default interface method

- In other words, by use of a default method, it is possible for an interface method to provide a body, rather than being abstract
- During its development, the default method was also referred to as an extension method, and you will likely see both terms used.
- A primary motivation for the default method was to provide a means by which interfaces could be expanded without breaking existing code

Example

ipse IDE
/indow Help

Light.java | Scannerint.java | testiface.java | stack.java | queue.java | sort.java | shape.java | NestedIFDem

```java
package mypkg;
interface ifdemo{
    int power(int base, int exponent);
    default int gcd(int n1, int n2){
        if(n2!=0) {
            return gcd(n2, n1%n2);
        }
        else {
            return n1;
        }
    }
}
class p implements ifdemo{
    public int power(int base, int exponent) {
        if(exponent==1) {
            return base;
        }
        else {
            return base*power(base,exponent-1);
        }

    }
}
public class defaultmethoddemo {

    public static void main(String[] args) {
        ifdemo demo= new p();
        System.out.println(demo.power(2, 5));
        System.out.println(demo.gcd(366, 60));// TODO Auto-generated method stub

    }

}
```

Problems | Javadoc | Declaration | Console

```
<terminated> defaultmethoddemo [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 17, 2019, 9:49:15 PM)
32
6
```

Static in interface

- Another capability added to interface by JDK 8 is the ability to define one or more static methods
- Like static methods in a class, a static method defined by an interface can be called independently of any object
- Thus, no implementation of the interface is necessary, and no instance of the interface is required, in order to call a static method

Example

:lipse IDE

'indow Help

Light.java | Scannerint.java | testiface.java | stack.java | queue.java | sort.java | shape.java | NestedIFDem...

```java
package mypkg;
interface ifdemo{
    int power(int base, int exponent);
    static int gcd(int n1, int n2){
        if(n2!=0) {
            return gcd(n2, n1%n2);
        }
        else {
            return n1;
        }
    }
}
class p implements ifdemo{
    public int power(int base, int exponent) {
        if(exponent==1) {
            return base;
        }
        else {
            return base*power(base,exponent-1);
        }

    }
}
public class defaultmethoddemo {

    public static void main(String[] args) {
        ifdemo demo= new p();
        System.out.println(demo.power(2, 5));
        System.out.println(ifdemo.gcd(366, 60));// TODO Auto-generated method stub

    }

}
```

Problems | @ Javadoc | Declaration | Console

<terminated> defaultmethoddemo [Java Application] C:\Program Files\Java\jre1.8.0_161\bin\javaw.exe (Nov 17, 2019, 9:49:15 PM)

```
32
6
```

Example

clipse IDE
/indow Help

Light.java | Scannerint.java | testiface.java | stack.java | queue.java | sort.java | shape.java | NestedIFDem... | defaultmeth

```
1  package mypkg;
2  interface ifdemo{
3      int power(int base, int exponent);
4      static int gcd(int n1, int n2){
5          if(n2!=0) {
6              return gcd(n2, n1%n2);
7          }
8          else {
9              return n1;
10         }
11     }
12 }
13 class p implements ifdemo{
14     public int power(int base, int exponent) {
15         if(exponent==1) {
16             return base;
17         }
18         else {
19             return base*power(base,exponent-1);
20         }
21
22     }
23 }
24 public class defaultmethoddemo {
25
26     public static void main(String[] args) {
27         ifdemo demo= new p();
28         System.out.println(demo.power(2, 5));
29         System.out.println(demo.gcd(366, 60));// TODO Auto-generated method stub
30
31     }
32
33 }
34
```

Problems | Javadoc | Declaration | Console

6 errors, 19 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (6 items)				
Abstract methods do not specify a body	abstractdemo...	/ICE107/src	line 2	Java Problem
Cannot override the final method from A6	finalmethod.j...	/ICE107/src	line 8	Java Problem
The method callme() is undefined for the type A	abstractdemo...	/ICE107/src	line 19	Java Problem
The method callmetoo() is undefined for the type A	abstractdemo...	/ICE107/src	line 20	Java Problem
The type B7 cannot subclass the final class A7	finalinheritan...	/ICE107/src	line 6	Java Problem
This static method of interface ifdemo can only be accessed as ifdemo.gcd	defaultmetho...	/ICE107/src/mypkg	line 29	Java Problem
Warnings (19 items)				