12 Inheritance and Exceptions

12.1

Packages are quite tricky, as your experimentation will show. The easiest way to experiment with them is to ensure that `CLASSPATH` includes ".", meaning that packages are subdirectories from whatever directory you are in when compiling. So here is class `A`:

```
1  package packageA;
2  public class A {
3    public static int x;
4    private static int y;
5    static int z;
6
7    public static void X() {
8      System.out.println(x+y+z);
9    }
10 }
```

If any class is to refer to class `A`, then `A` must be locatable in subdirectory `packageA`, starting from some directory in the `CLASSPATH`. So we assume that file `A.java` is located in subdirectory `packageA`.

Now we want another class, `B`, to be in a different package, but in subdirectory `packageA`; class `B` is going to refer to the static variables of class `A` (and we'll see that some of the references are not permitted). But first, let's consider a simple version of `B` that refers only to the public variable `x`:

```
1  package packageB;
```

```
2  public class simpleB {
3    public void X() {
4      int a=A.x;
5    }
6  }
```

If we are situated in subdirectory `packageA`, and we type `javac simpleB.java`, we get:

```
simpleB.java:4: cannot resolve symbol
symbol  : variable A
location: class packageB.simpleB
    int a=A.x;
          ^
1 error
```

The Java compiler is unable to resolve the symbol `A` as a class name. This is because class `A` was compiled in package `packageA`, so it actually has the name `packageA.A`. So we should be able to modify the reference as in the following class, `simpleC`:

```
1  package packageB;
2  public class simpleC {
3    public void X() {
4      int a=packageA.A.x;
5    }
6  }
```

This time we get a different compilation error:

```
simpleC.java:4: cannot resolve symbol
symbol  : class A
location: package packageA
    int a=packageA.A.x;
                   ^
1 error
```

The reason for this is that we are compiling *from* the directory `packageA`; when the compiler goes looking for the reference, it looks for variable `x` in class `A` in package `packageA`, which means in *subdirectory* `packageA`. But we're already *in* subdirectory `packageA`. The solution is simple – we must move to the directory *above* `packageA`, and then type `javac packageA/simpleC.java`. Everything then works fine. Incidentally, the more common way of writing class `simpleC` would be to use an `import packageA` statement.

So back to the original problem. We want a class `B` in file `packageA/B.java` as follows:

```
1  package packageB;
2  import packageA.*;
3  public class B {
4    public void X() {
5      int a=A.x,
```

```
6           b=A.y,
7           c=A.z;
8     }
9   }
```

We have used an `import packageA;` statement, and we know that we must compile from the directory above `packageA`. But we get

```
packageA/B.java:6: y has private access in packageA.A
        b=A.y,
           ^
packageA/B.java:7: z is not public in packageA.A; cannot be accessed from outside package
        c=A.z;
           ^
2 errors
```

The errors are self-explanatory. Variable `y` is private, so no other class can acces it; variable `z` has package visibility in package `packageA`, so it cannot be accessed from a class that is in `packageB`.

12.2

```
1   public class MilitaryClock extends Clock {
2     // Clock with military time
3     // Janet Kamin, August 20, 2001
4
5     public MilitaryClock () {}
6
7     public void advanceMinutes (int m) {
8       int totalMinutes = (hour * 60 + m) % (24 * 60);
9       // totalMinutes is between -(24 * 60) and (24 * 60)
10      if (totalMinutes < 0) totalMinutes = totalMinutes + (24 * 60);
11      hour = totalMinutes / 60;
12      if (hour == 0) hour = 12;
13      minute = totalMinutes % 60;
14    }
15  }
```

12.3

```
public PreciseClock (int h, int m, int s) {
  super(h, m);
  second = s;
}
```

12.4

This algorithm doesn't work; the mouse runs back and forth forever along the northern corridor.

12.5

(Nothing interesting to show.)

12.6

```
1  public class LeftMouse extends Mouse {
2    // A mouse that can navigate a maze, using the
3    // "wall-to-the-left" algorithm
4    // Author: Allan M. Mickunas, September 22, 1996
5
6    public LeftMouse (Maze m) {
7      super(m);
8    }
9
10   public void makeMove () {
11   // Use "wall-to-the-right" algorithm
12     if (started) {
13       if (!outside()) {
14         turnLeft();
15         while (facingWall()) {
16           turnRight();
17         }
18         stepForward();
19       }
20     } else {
21       stepForward();
22       started=true;
23     }
24   }
25 }
26
27
```

and

```
1  import CSLib.*;
2
3  public class LeftMouseController {
4    // Draw a mouse navigating a maze, using the
5    // "hug the wall to the right" algorithm
6    // Rebecca Kamin, Sept 12, 2000
7
8    public void runMouse () {
```

```
    DrawingBox d = new DrawingBox("Mouse in maze");
    Maze theMaze = new Maze();
    MazeDrawer theMazeDrawer = new MazeDrawer(theMaze, d);
    LeftMouse speedy = new LeftMouse(theMaze);
    MouseDrawer speedyDrawer =
                  new MouseDrawer(speedy, theMazeDrawer, d);

    while (true) {
      theMazeDrawer.draw();
      speedyDrawer.draw();
      Timer.pause(1000);
      speedy.makeMove();
    }
  }
}
```

12.7

```
import CSLib.*;

public class MouseController {
  // Draw a mouse navigating a maze, using a
  // choice of algorithms.
  // Rebecca Kamin, Sept 12, 2000

  public void runMouse () {
    DrawingBox d = new DrawingBox("Mouse in maze");
    Maze theMaze = new Maze();
    MazeDrawer theMazeDrawer = new MazeDrawer(theMaze, d);

    InputBox mouseChoice = new InputBox();
    mouseChoice.setPrompt("Choose RightMouse (0) or StraightMouse (1) or LeftMouse (2)");
    int choice = mouseChoice.readInt();

    Mouse speedy;
    switch (choice) {
      case 0:  speedy = new RightMouse(theMaze);
               break;
      case 1:  speedy = new StraightMouse(theMaze);
               break;
      default: speedy = new LeftMouse(theMaze);
               break;
    }

    MouseDrawer speedyDrawer =
                  new MouseDrawer(speedy, theMazeDrawer, d);

    while (true) {
      theMazeDrawer.draw();
      speedyDrawer.draw();
```

```
33          Timer.pause(1000);
34          speedy.makeMove();
35        }
36      }
37    }
```

12.8

One possibility is to add a new public instance variable to class `Operation` – say `boolean exceptionFound`. Then add some code to the `calculate` method to check whether `opnd2==0` during division, and if the dividend is zero, set `exceptionFound` to `true` (and don't perform the division). Then any call to `op.calculate()` should be followed by a test to see whether `op.exceptionFound` is set to `true`.

12.9

Input `40+x` throws a `NumberFormatException`. Input `50&43` returns 0 (the "shouldn't happen" case of the switch statement).

12.10

```
1   import CSLib.*;
2
3   public class Exceptions {
4     OutputBox out = new OutputBox();
5     InputBox in = new InputBox();
6
7     public int throwae () {
8       boolean b = true;
9       in.setPrompt("Enter 0 to cause Arithmetic Exception");
10      int x = in.readInt();
11      if (b)
12        return 20/x;
13      else
14        return 2;
15    }
16
17    public void throwaioobe () {
18      int sum = 0;
19      in.setPrompt("Enter 4 to cause Array Index Out Of Bounds Exception");
20      int n = in.readInt();
21      int[] A = new int[3];
22      for (int i=1; i<n; i++)
23        sum = sum + A[i];
24    }
25
26    public void thrownase () {
27      int[] A;
28      in.setPrompt("Enter negative to cause Negative Array Size Exception");
29      int i = in.readInt();
```

```
      A=new int[i];
    }

    public void thrownpe () {
      OutputBox[] A = new OutputBox[1];
      for (int i=0; i<1; i++)
        A[i].println();
    }
  }
```

```
  import CSLib.*;

  public class CaughtExceptions {
    OutputBox out = new OutputBox();
    InputBox in = new InputBox();

    public int throwae () {
      boolean b = true;
      in.setPrompt("Enter 0 to cause Arithmetic Exception");
      int x = in.readInt();
      try {
        if (b)
          return 20/x;
        else
          return 2;
      } catch (ArithmeticException e) {
        out.println("Caught Arithmetic Exception");
        return 0;
      }
    }

    public void throwaioobe () {
      int sum = 0;
      in.setPrompt("Enter 4 to cause Array Index Out Of Bounds Exception");
      int n = in.readInt();
      int[] A = new int[3];
      try {
        for (int i=1; i<n; i++)
          sum = sum + A[i];
      } catch (ArrayIndexOutOfBoundsException e) {
        out.println("Caught Array Index Out Of Bounds Exception");
      }
    }

    public void thrownase () {
      int[] A;
      in.setPrompt("Enter negative to cause Negative Array Size Exception");
      int i = in.readInt();
      try {
        A=new int[i];
```

```
41      } catch (NegativeArraySizeException e) {
42        out.println("Caught Negative Array Size Exception");
43      }
44    }
45
46    public void thrownpe () {
47      OutputBox[] A = new OutputBox[1];
48      try {
49      for (int i=0; i<1; i++)
50        A[i].println();
51      } catch (NullPointerException e) {
52        out.println("Caught Null Pointer Exception");
53      }
54    }
55  }
```

12.11

```
1   import CSLib.*;
2
3   public class CaughtExceptions {
4     OutputBox out = new OutputBox();
5
6     public int throwae () {
7       try {
8         return extra();
9       } catch (ArithmeticException e) {
10        out.println("Caught Arithmetic Exception");
11        return 0;
12      }
13    }
14
15    private int extra () {
16      int x = 0;
17      boolean b = true;
18      if (b)
19        try {
20          return 20/x;
21        } catch (ArithmeticException e) {
22          throw e;
23        }
24      else
25        return 2;
26    }
27  }
```

12.12

```
1   import CSLib.*;
2
```

```
3   public class CaughtExceptions {
4     OutputBox out1 = new OutputBox("Number 1"),
5               out2 = new OutputBox("Number 2");
6     InputBox in = new InputBox();
7
8     public int throwae () {
9       try {
10        return extra();
11      } catch (ArithmeticException e) {
12        out1.println("Caught Arithmetic Exception");
13        return 0;
14      }
15    }
16
17    private int extra () {
18      in.setPrompt("Enter 0 to cause Arithmetic Exception");
19      out2.println("Extra is writing on 2");
20      int x = in.readInt();;
21      boolean b = true;
22      if (b)
23        try {
24          return 20/x;
25        } catch (ArithmeticException e) {
26          throw e;
27        }
28        finally {
29          Timer.pause(5000);
30          out2.dispose();
31        }
32      else
33        return 2;
34    }
35  }
```

12.13

```
1   import CSLib.*;
2
3   public class Exceptions {
4     InputBox in = new InputBox();
5
6     public int throwae () {
7       boolean b = true;
8       in.setPrompt("Enter 0 to cause Arithmetic Exception");
9       int x = in.readInt();
10      if (b)
11        return 20/x;
12      else
13        return 2;
14    }
```

```

  public void throwaioobe () {
    int sum = 0;
    in.setPrompt("Enter 4 to cause Array Index Out Of Bounds Exception");
    int n = in.readInt();
    int[] A = new int[3];
    for (int i=1; i<n; i++)
      sum = sum + A[i];
  }

  public void thrownase () {
    int[] A;
    in.setPrompt("Enter negative to cause Negative Array Size Exception");
    int i = in.readInt();
    A=new int[i];
  }

  public void thrownpe () {
    OutputBox[] A = new OutputBox[1];
    for (int i=0; i<1; i++)
      A[i].println();
  }
}
```

```
import CSLib.*;

public class CaughtExceptionsClient {

  public static void main (String[] args) {
    OutputBox out = new OutputBox();
    Exceptions e = new Exceptions();
    try {
      e.throwae();
      e.throwaioobe();
      e.thrownase();
      e.thrownpe();
      }
    catch (ArithmeticException e1) {
      out.println("Caught Arithmetic Exception");
    }
    catch (ArrayIndexOutOfBoundsException e2) {
      out.println("Caught Array Index Out Of Bounds Exception");
    }
    catch (NegativeArraySizeException e3) {
      out.println("Caught Negative Array Size Exception");
    }
    catch (NullPointerException e4) {
      out.println("Caught Null Pointer Exception");
    }
  }
```

```
27    }
```

12.14

En tering4+ causes a run-time error:

```
Running CalculatorClient
java.lang.StringIndexOutOfBoundsException: String index out of range: 3
        at java.lang.String.substring(String.java:1500)
        at Operation.getInteger(Operation.java:37)
        at Operation.parse(Operation.java:24)
        at Calculator.calcloop(Calculator.java:15)
        at CalculatorClient.main(CalculatorClient.java:7)
```

One solution is to append a blank to the end of the input string, just after it is passed into `parse`.

```
  static Operation parse (String s) {
    s = s + " ";
    // s should have form "integer op integer"
    // Divide into three parts and return Operation object

    int i = getInteger(s, 0);
    int opnd1 = Integer.parseInt(inputfrag);

    char optr = s.charAt(i);

    i = getInteger(s, i+1);
    int opnd2 = Integer.parseInt(inputfrag);

    return new Operation(opnd1, opnd2, optr);
  }
```