APPinc.
Making Technology Reachable

Java Programming

RITA M. BARRIOS, Ph.D.

PRES. TRAINING & EDUCATION

Let’s look at the code

We use command line arguments here

- public static void main(String[] **arguments**)
- The first is the month (1-12)
- The second is the year (CCYY)
- If we omit the command line arguments, it will use 1/2012

Lines 13 -40: Switch Statement to count the days in the month

```
static int countDays(int month, int year) {
        int count = -1;
        switch (month) {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
            case 12:
                count = 31;
                break;
            case 4:
            case 6:
            case 9:
            case 11:
                count = 30;
                break;
            case 2:
                if (year % 4 == 0)
                    count = 29;
                else
                    count = 28;
                if ((year % 100 == 0) & (year % 400 != 0))
                    count = 28;
        }
        return count;
    }
```

DayCounter.java – Command Line

Right click on DayCounter.java

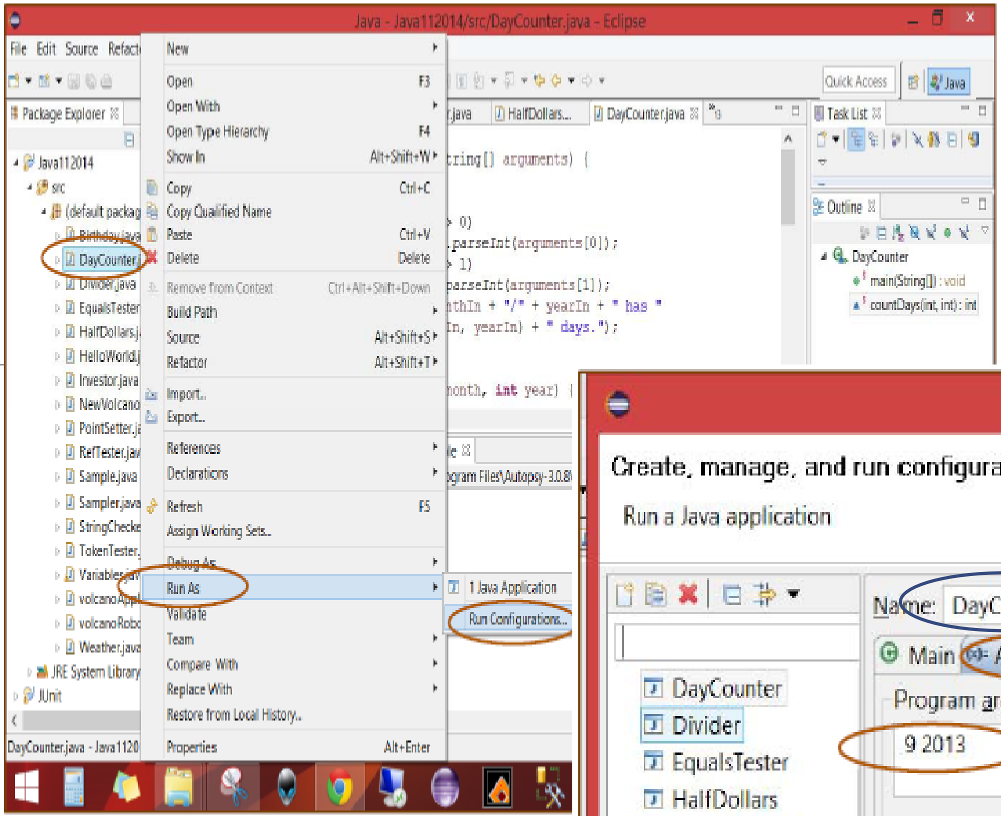
Select Run As

Run Configurations

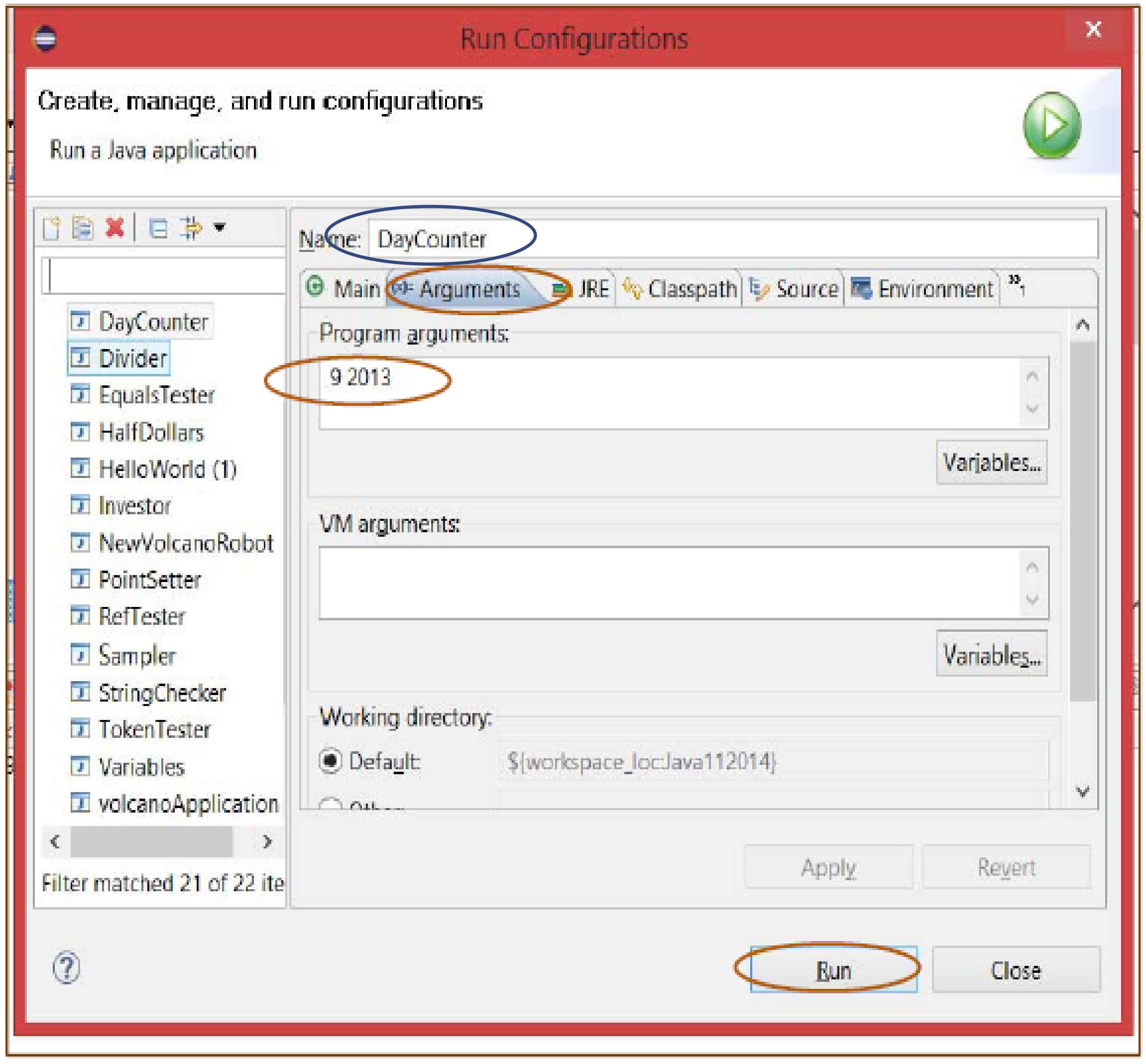
Select the Arguments Tab

Enter a Month and full year (CCYY) separated by a space in the arguments field

Click Run



```
tring[] arguments) {
 0)
.parseInt(arguments[0]);
 1)
parseInt(arguments[1]);
nthIn + "/" + yearIn + " has "
In, yearIn) + " days.");
nonth, int year) {
```



Name: DayCounter

Program arguments:

9 2013

VM arguments:

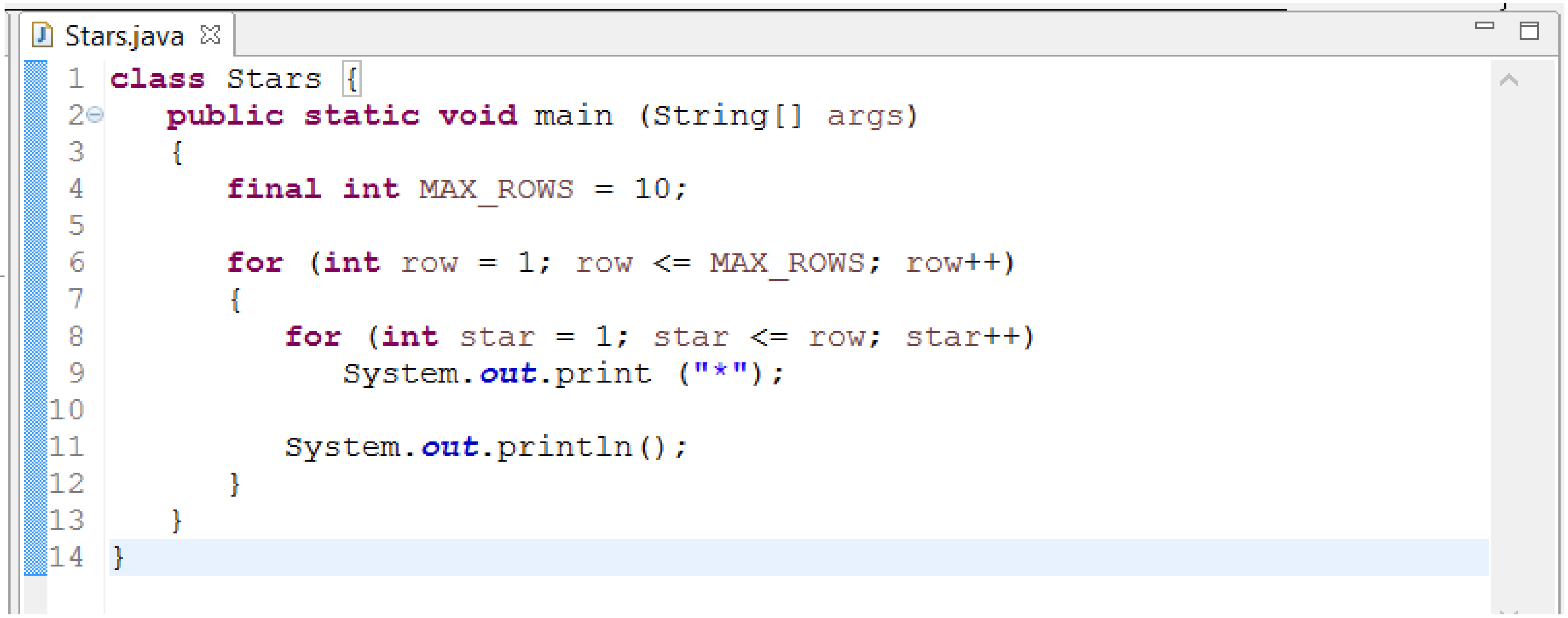
Working directory:

Default: ${workspace_loc:Java112014}

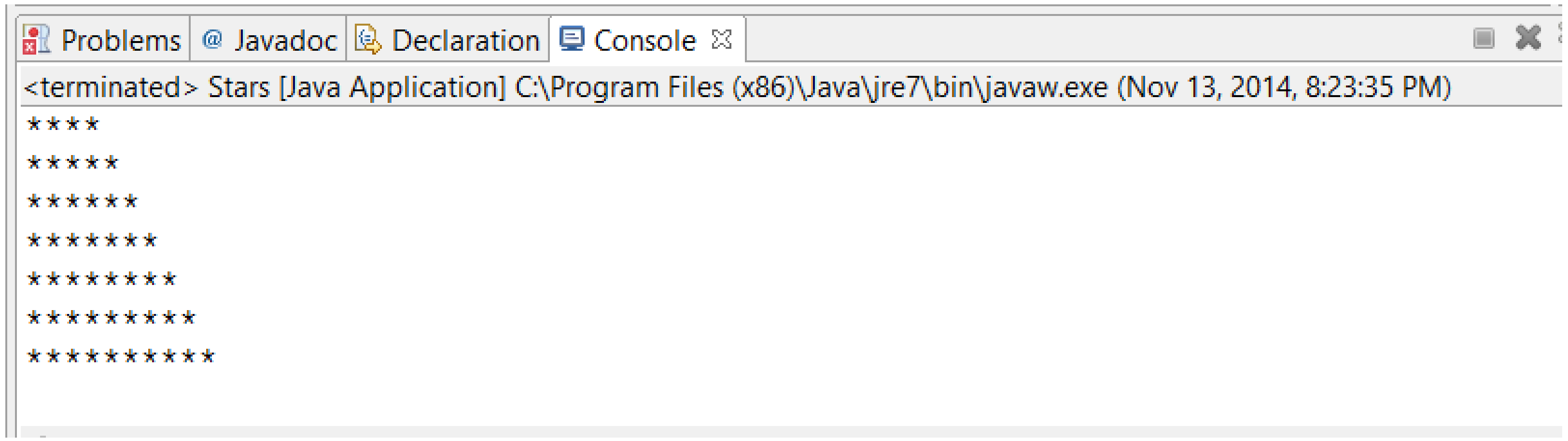
Loop Exercise

Create a file called Stars.java. This code will print out a triangle using the *.

Run as a Java Application



```java
class Stars {
   public static void main (String[] args)
   {
      final int MAX_ROWS = 10;

      for (int row = 1; row <= MAX_ROWS; row++)
      {
         for (int star = 1; star <= row; star++)
            System.out.print ("*");

         System.out.println();
      }
   }
}
```



```
<terminated> Stars [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 13, 2014, 8:23:35 PM)
****
*****
******
*******
********
*********
**********
```

Drawing complex figures

Use nested `for` loops to produce the following output.

```
#================#
|      <><>      |
|    <>....<>    |
|  <>........<>  |
|<>............<>|
|<>............<>|
|  <>........<>  |
|    <>....<>    |
|      <><>      |
#================#
```

Development strategy

Recommendations for managing complexity:

1. Write an English description of steps required (*pseudo-code*)
    - use pseudo-code to decide methods

2. Create a table of patterns of characters
    - use table to write loops in each method

```
#================#
|      <><>      |
|    <>....<>    |
|  <>........<>  |
|<>............<>|
|<>............<>|
|  <>........<>  |
|    <>....<>    |
|      <><>      |
#================#
```

1. Pseudo-code

pseudo-code: An English description of an algorithm.

Example: Drawing a 12 wide by 7 tall box of stars

```
print 12 stars.
for (each of 5 lines) {
    print a star.
    print 10 spaces.
    print a star.
}
print 12 stars.
```

```
************
*          *
*          *
*          *
*          *
*          *
************
```

Pseudo-code algorithm

1. Line 1
   1 hash (#) sign , 16 equal (=) signs and 1 equal (#) sign

2. Top half
   - |
   - spaces (decreasing)
   - <>
   - dots (increasing)
   - <>
   - spaces (same as above)
   - |

3. Bottom half (top half upside-down)

4. Last Line
   1 hash (#) sign , 16 equal (=) signs and 1 equal (#) sign

```
#================#
|      <><>      |
|    <>....<>    |
|  <>........<>  |
|<>............<>|
|<>............<>|
|  <>........<>  |
|    <>....<>    |
|      <><>      |
#================#
```

Methods from pseudo code

```
public class Mirror {
    public static void main(String[] args) {
        line();
        topHalf();
        bottomHalf();
        line();
    }

    public static void topHalf() {
        for (int line = 1; line <= 4; line++) {
            // contents of each line
        }
    }

    public static void bottomHalf() {
        for (int line = 1; line <= 4; line++) {
            // contents of each line
        }
    }

    public static void line() {
        // ...
    }
}
```

2. Tables

A table for the top half:

- Compute spaces and dots expressions from line number

line	spaces	line * -2 + 8	dots	4 * line - 4
1	6	6	0	0
2	4	4	4	4
3	2	2	8	8
4	0	0	12	12

```
#================#
|      <><>      |
|    <>....<>    |
|  <>........<>  |
|<>............<>|
|<>............<>|
|  <>........<>  |
|    <>....<>    |
|      <><>      |
#================#
```

3. Writing the code

Useful questions about the top half:

- What methods can we use?
  - (think structure and redundancy)
- Number of (nested) loops per line?

```
#================#
|      <><>      |
|    <>....<>    |
|  <>........<>  |
|<>............<>|
|<>............<>|
|  <>........<>  |
|    <>....<>    |
|      <><>      |
#================#
```

Partial solution – You finish the rest..

```java
public class Mirror {
      // Prints the expanding pattern of <> for the top half of the figure.
      public static void main(String [] args) {
            line();
            topHalf();
            bottomHalf();
            line();
      }

      // Prints the expanding pattern of <> for the top half of the figure.
      public static void topHalf() {
          for (int line = 1; line <= 4; line++) {
              System.out.print("|");

              for (int space = 1; space <= (line * -2 + 8); space++) {
                  System.out.print(" ");
              }

              System.out.print("<>");

              for (int dot = 1; dot <= (line * 4 - 4); dot++) {
                  System.out.print(".");
              }

              System.out.print("<>");

              for (int space = 1; space <= (line * -2 + 8); space++) {
                  System.out.print(" ");
              }

              System.out.println("|");
          }
      }

    public static void bottomHalf() {
        for (int line = 1; line <= 4; line++) {
            // contents of each line
        }
    }

    public static void line() {
        // ...
    }
}
```

```
#================#
|      <><>      |
|    <>....<>    |
|  <>........<>  |
|<>............<>|
|<>............<>|
|  <>........<>  |
|    <>....<>    |
|      <><>      |
#================#
```

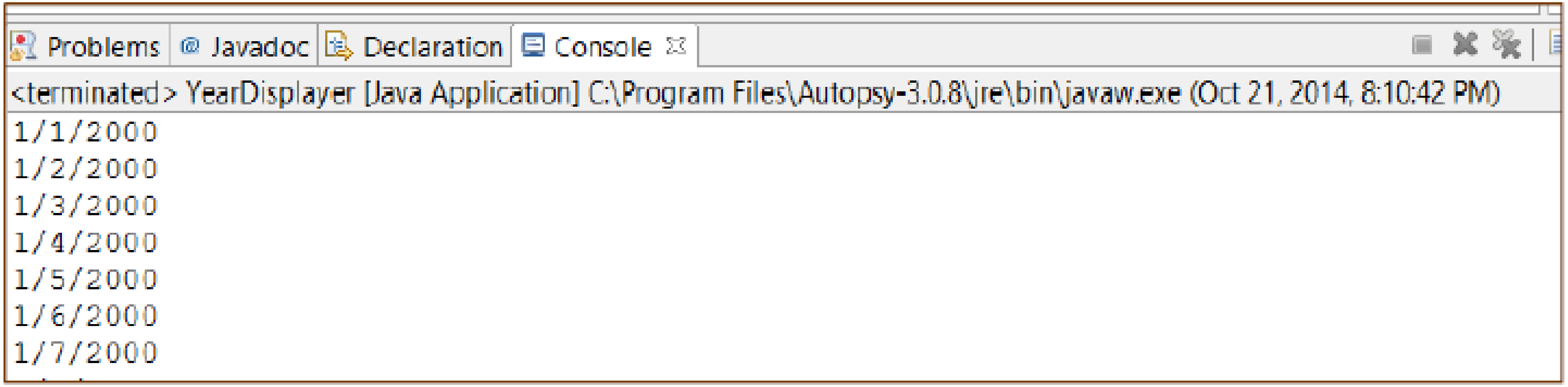
Independent Exercise Revisited – YearDisplayer.java

A. Using countDays() method from the DayCounter.java application, create an application that displays every date in a given year in a single list from January 1 to December 31.

Name the Java file as YearDisplayer.java

Run as a Java Application

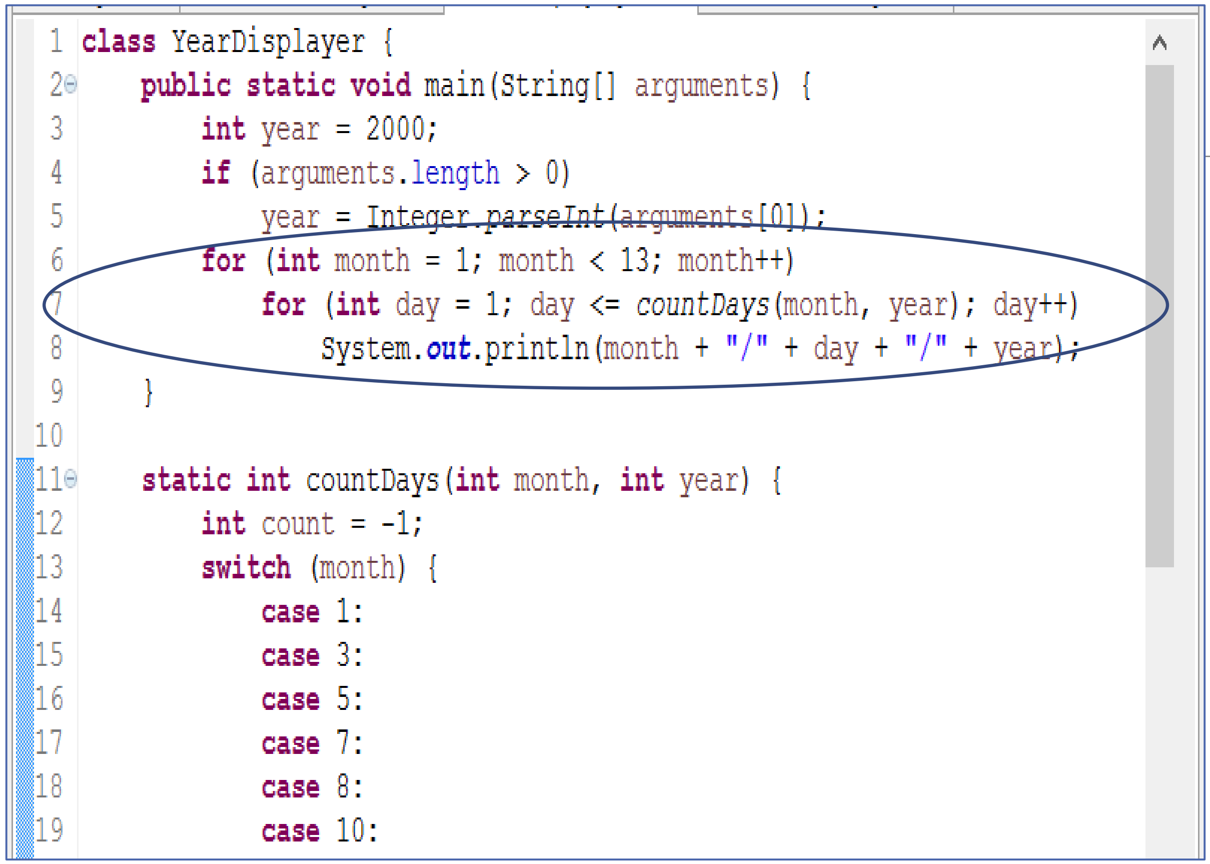
YearDisplayer.java (partial output)



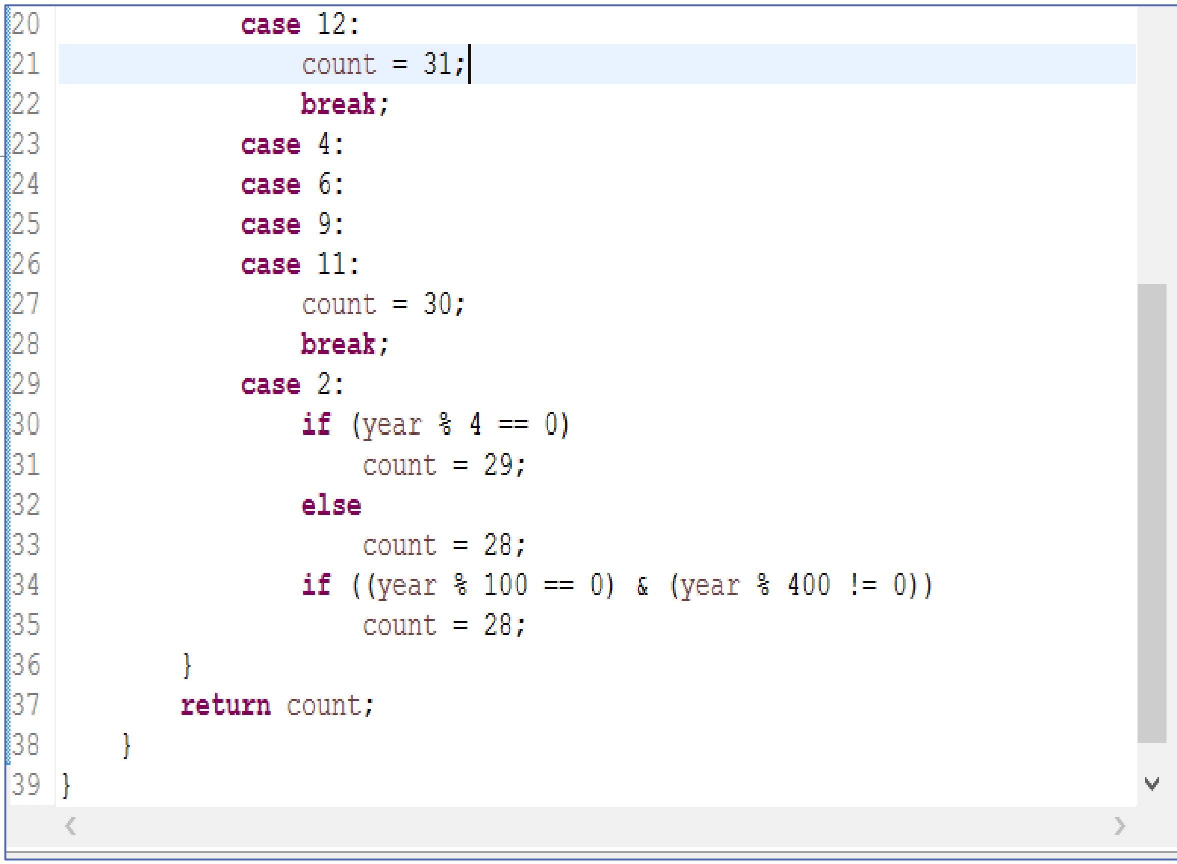
```
<terminated> YearDisplayer [Java Application] C:\Program Files\Autopsy-3.0.8\jre\bin\javaw.exe (Oct 21, 2014, 8:10:42 PM)
1/1/2000
1/2/2000
1/3/2000
1/4/2000
1/5/2000
1/6/2000
1/7/2000
```

Solution

YearDisplayer.java



```java
class YearDisplayer {
    public static void main(String[] arguments) {
        int year = 2000;
        if (arguments.length > 0)
            year = Integer.parseInt(arguments[0]);
        for (int month = 1; month < 13; month++)
            for (int day = 1; day <= countDays(month, year); day++)
                System.out.println(month + "/" + day + "/" + year);
    }

    static int countDays(int month, int year) {
        int count = -1;
        switch (month) {
            case 1:
            case 3:
            case 5:
            case 7:
            case 8:
            case 10:
```



```java
                case 12:
                    count = 31;
                    break;
                case 4:
                case 6:
                case 9:
                case 11:
                    count = 30;
                    break;
                case 2:
                    if (year % 4 == 0)
                        count = 29;
                    else
                        count = 28;
                    if ((year % 100 == 0) & (year % 400 != 0))
                        count = 28;
            }
            return count;
        }
}
```

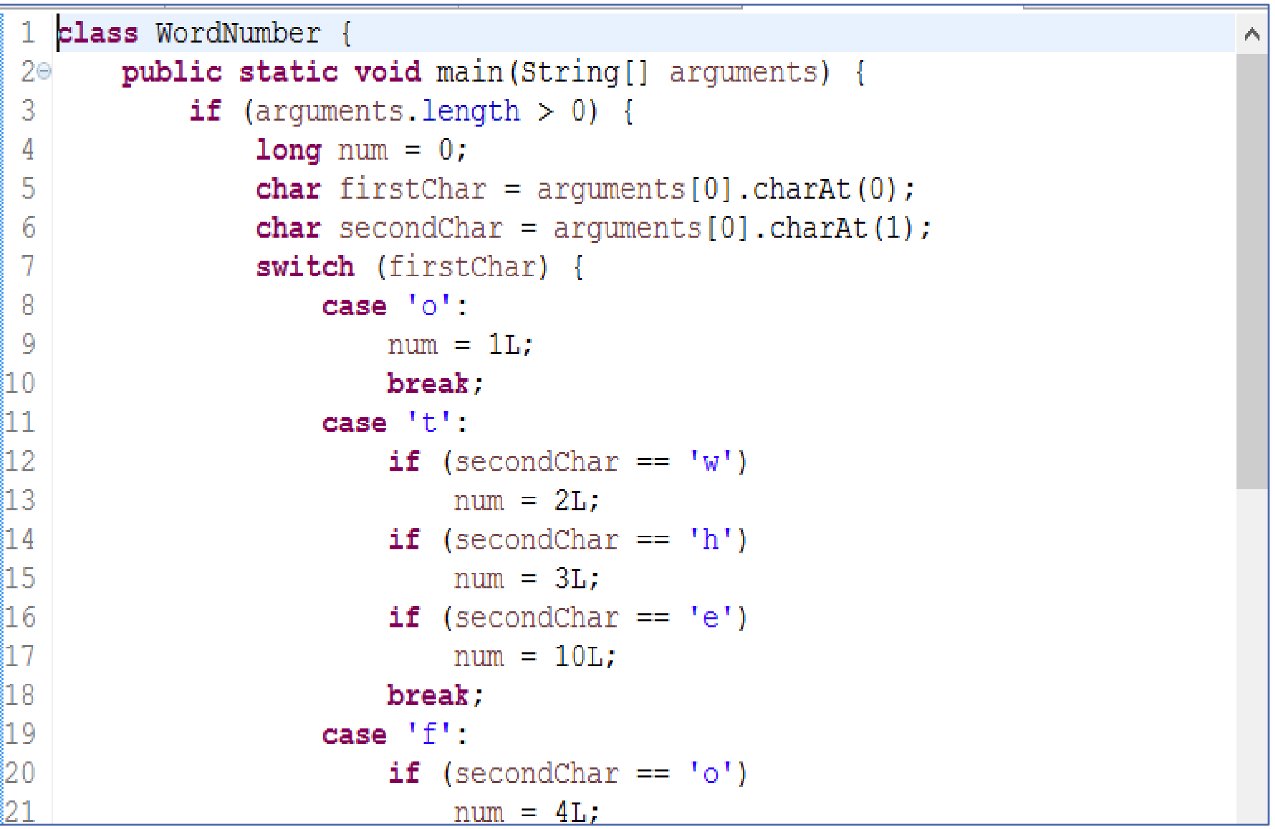
Independent Exercise Revisited – WordNumber.java

B. Create a class that takes words from the first 10 numbers (“one” – “ten”) and converts them into a single long integer. Use the switch Statement for the conversion and command-line arguments for the words.

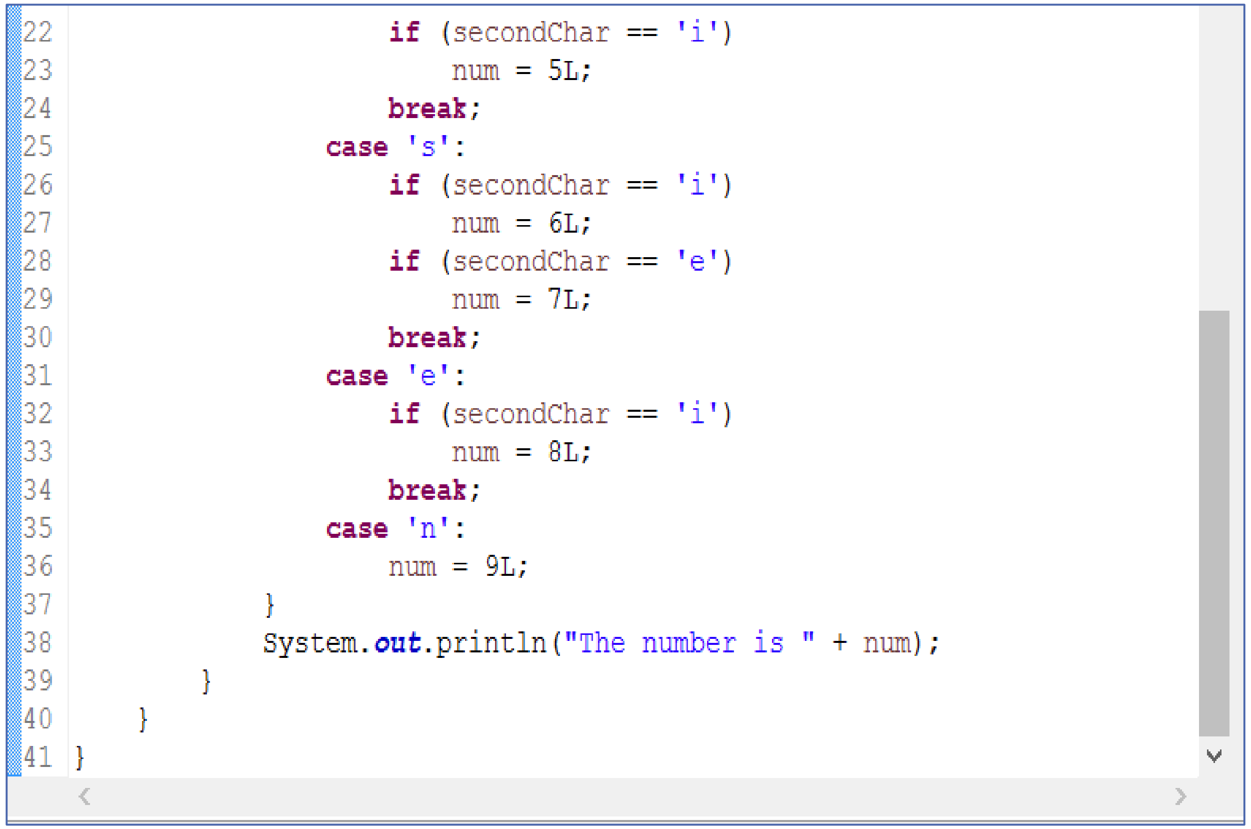
Name the file WordNumber.java

Run as command line with your word selection (shown next)

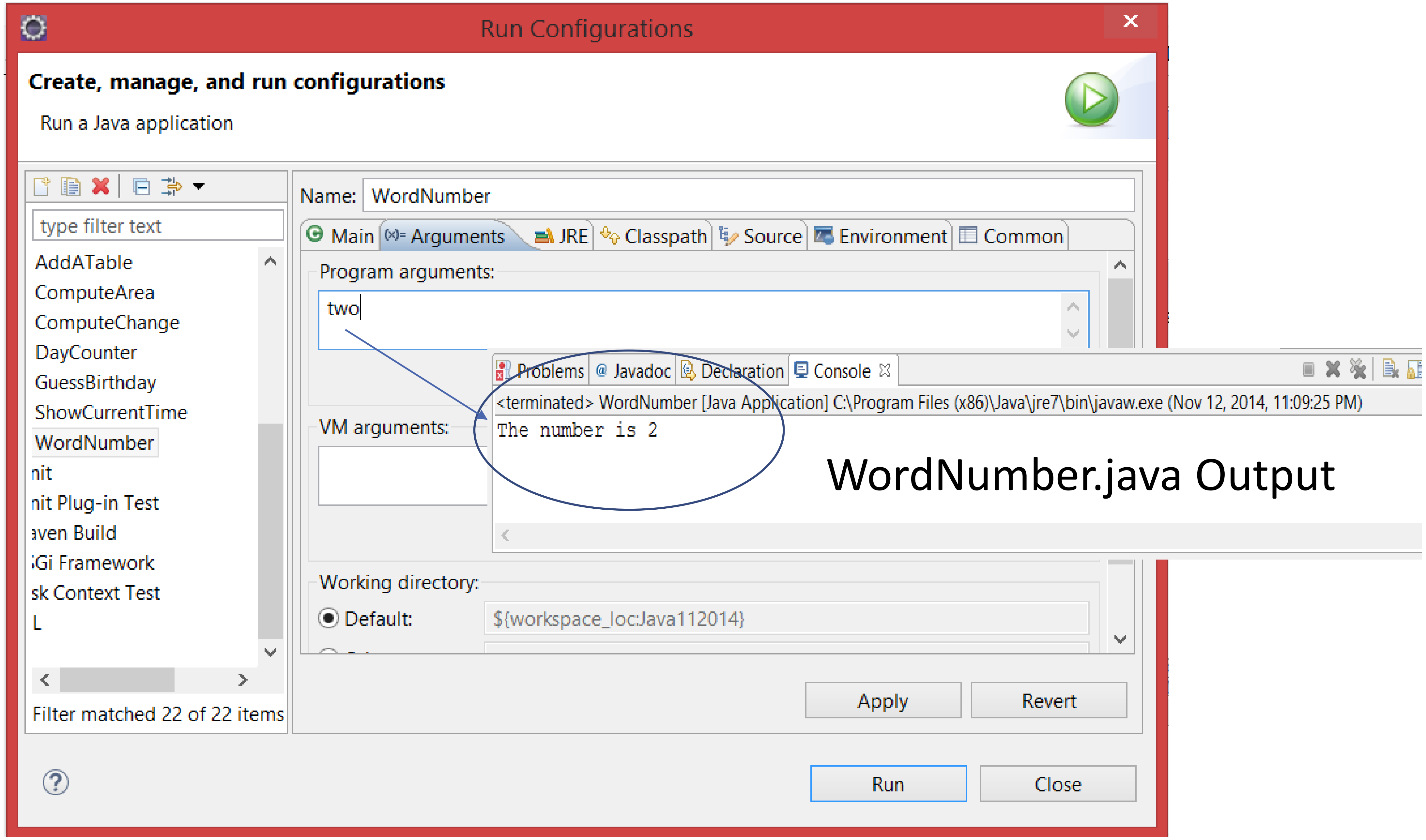
WordNumber.java Solution
Run as with Command-line Arguments



```java
class WordNumber {
    public static void main(String[] arguments) {
        if (arguments.length > 0) {
            long num = 0;
            char firstChar = arguments[0].charAt(0);
            char secondChar = arguments[0].charAt(1);
            switch (firstChar) {
                case 'o':
                    num = 1L;
                    break;
                case 't':
                    if (secondChar == 'w')
                        num = 2L;
                    if (secondChar == 'h')
                        num = 3L;
                    if (secondChar == 'e')
                        num = 10L;
                    break;
                case 'f':
                    if (secondChar == 'o')
                        num = 4L;
```



```java
                    if (secondChar == 'i')
                        num = 5L;
                    break;
                case 's':
                    if (secondChar == 'i')
                        num = 6L;
                    if (secondChar == 'e')
                        num = 7L;
                    break;
                case 'e':
                    if (secondChar == 'i')
                        num = 8L;
                    break;
                case 'n':
                    num = 9L;
            }
            System.out.println("The number is " + num);
        }
    }
}
```



Name: WordNumber

Program arguments:

```
two
```

VM arguments:

Working directory:

Default: ${workspace_loc:Java112014}

```
<terminated> WordNumber [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 12, 2014, 11:09:25 PM)
The number is 2
```

WordNumber.java Output

Exercise – Be the Compiler

Use the document on Google Group called MethodsAndInstanceVariablesExerciseA.pdf

You will be given two classes. Your job is to figure out if they compile. If they do, what is the output. If they don’t how would you fix it?

Solution Next Slide…

Exercise – Be the Compiler Solution

Class Xcopy compiles and runs as is. The output is 42.84. The go() method does not change the orig variable since Java is pass by value.

Class Clock – method geTime() should be corrected to change from void to String since it has a return of time, a string

```
String getTime() {
        return time;
}
```

Class Methods

Like other methods but available to any instance of the class itself as well as other classes

Does not require an object of the class for its methods to be called

For example, System class in the Java Class Library defines a set of methods that we can use when ever we want – a couple are shown here

System.exit(0); ← close an application when a status code indicates success

long now = **System.currentTimeMillis();** ← return a long holding the number of milliseconds since midnight 1/1/1970

System.out.println(); ← print a line to the console

Class Methods (2)

Defined with the keyword static in front of the method declaration – if omitted, it becomes an instance method

- Instance methods operate on objects instead of a class

Cannot be inherited

It is a superclass and cannot be overridden in a subclass

Method Overloading

Methods with the same name but different signatures

- Number of arguments are different
- Primitive type or objects of each argument is different

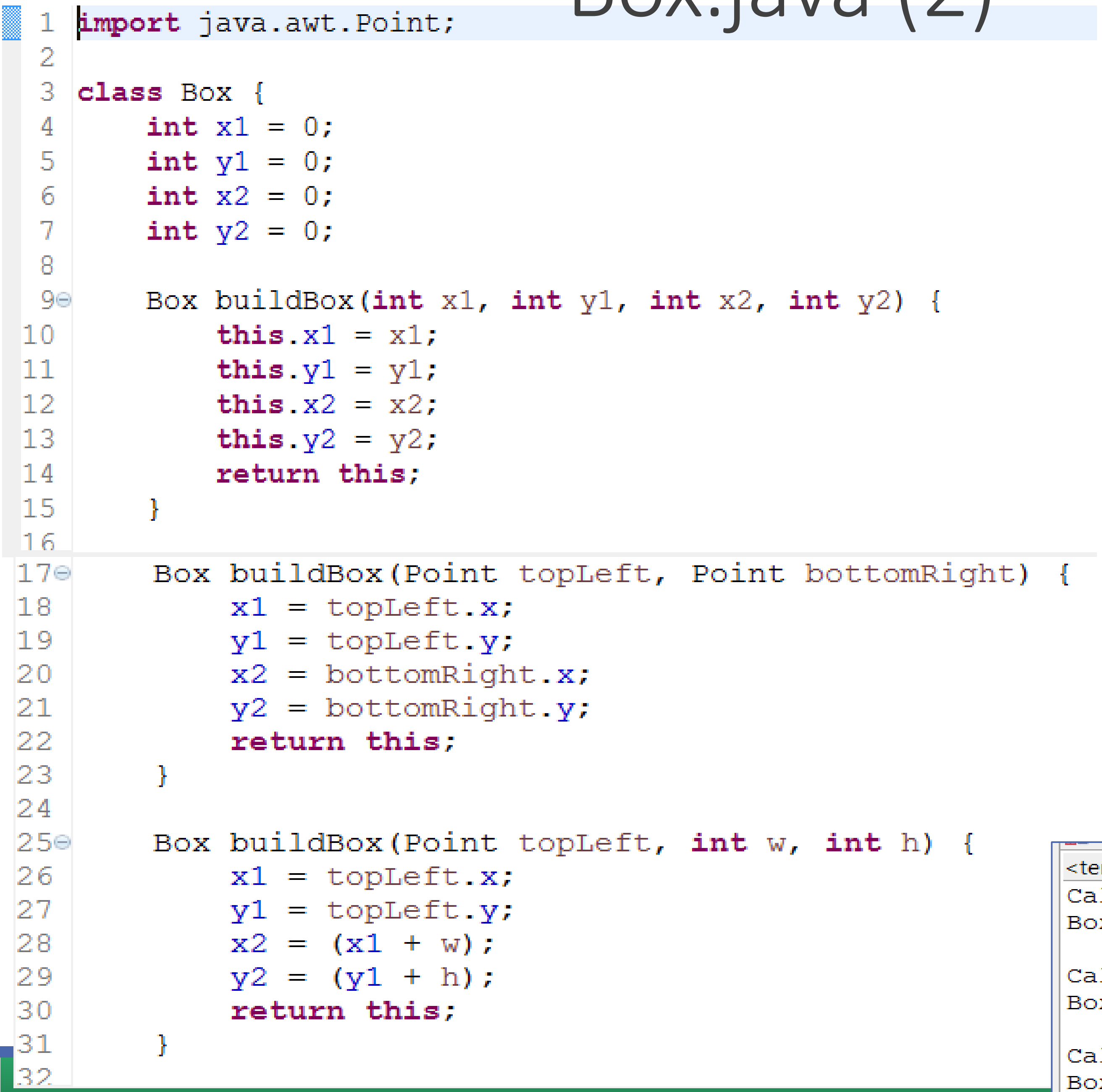
To create an overloaded method

- Create different method definitions in the class, each with same name but different argument list
  - Argument list must be different to avoid collisions

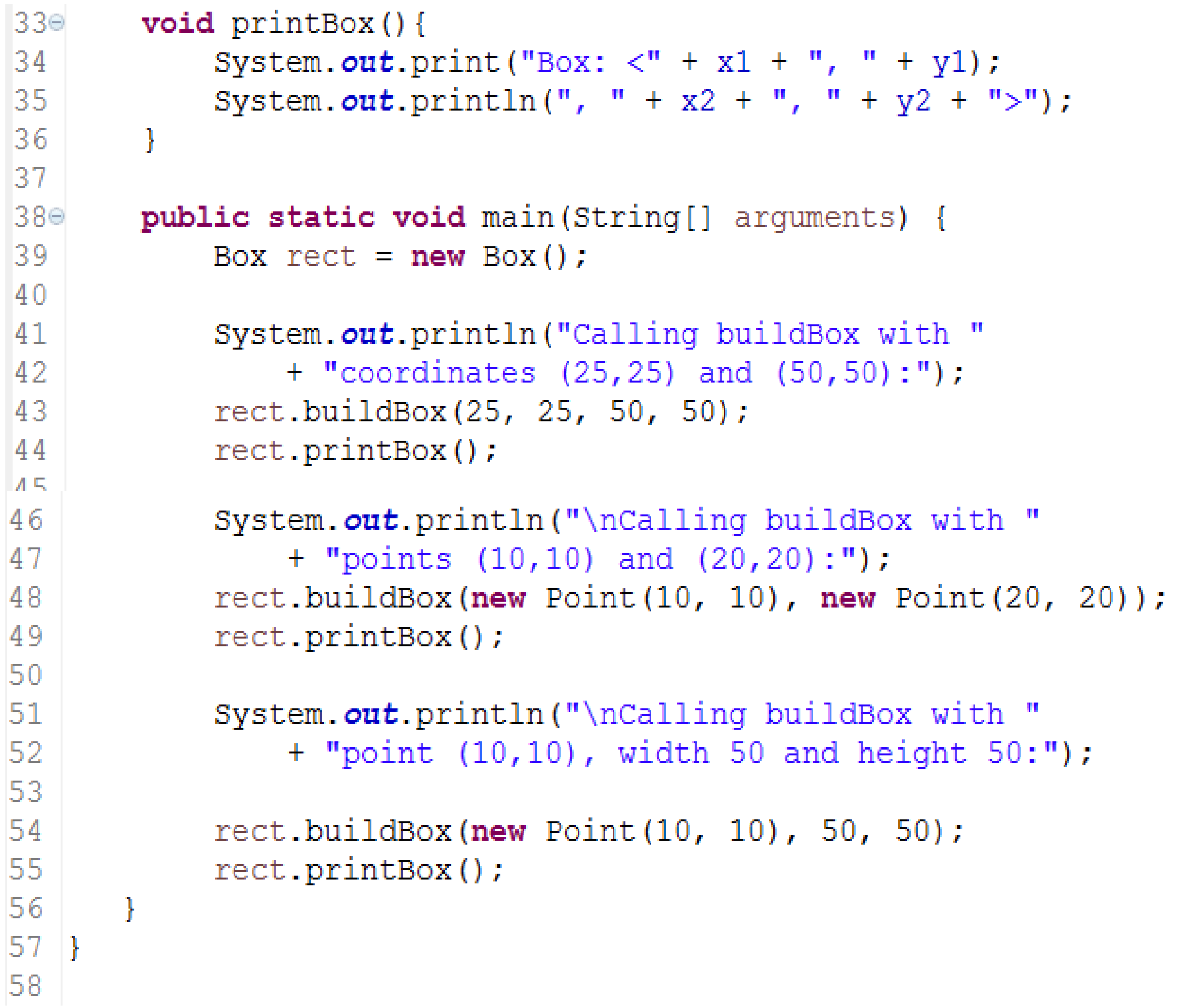
Exercise 1: Box.java – Page 106

Create an overloaded method beginning with a simple class called Box.java to define a rectangle with four instance variables to define the upper-left and lower-right bounds of the rectangle – (x1, y1) and (x2, y2)

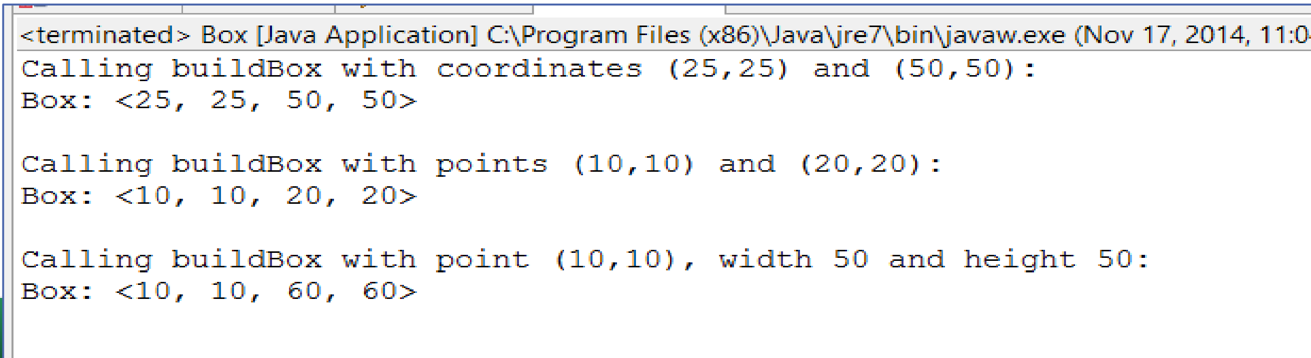
Box.java (2)



```java
import java.awt.Point;

class Box {
    int x1 = 0;
    int y1 = 0;
    int x2 = 0;
    int y2 = 0;

    Box buildBox(int x1, int y1, int x2, int y2) {
        this.x1 = x1;
        this.y1 = y1;
        this.x2 = x2;
        this.y2 = y2;
        return this;
    }

    Box buildBox(Point topLeft, Point bottomRight) {
        x1 = topLeft.x;
        y1 = topLeft.y;
        x2 = bottomRight.x;
        y2 = bottomRight.y;
        return this;
    }

    Box buildBox(Point topLeft, int w, int h) {
        x1 = topLeft.x;
        y1 = topLeft.y;
        x2 = (x1 + w);
        y2 = (y1 + h);
        return this;
    }

```



```java
    void printBox(){
        System.out.print("Box: <" + x1 + ", " + y1);
        System.out.println(", " + x2 + ", " + y2 + ">");
    }

    public static void main(String[] arguments) {
        Box rect = new Box();

        System.out.println("Calling buildBox with "
            + "coordinates (25,25) and (50,50):");
        rect.buildBox(25, 25, 50, 50);
        rect.printBox();

        System.out.println("\nCalling buildBox with "
            + "points (10,10) and (20,20):");
        rect.buildBox(new Point(10, 10), new Point(20, 20));
        rect.printBox();

        System.out.println("\nCalling buildBox with "
            + "point (10,10), width 50 and height 50:");

        rect.buildBox(new Point(10, 10), 50, 50);
        rect.printBox();
    }
}

```



```
<terminated> Box [Java Application] C:\Program Files (x86)\Java\jre7\bin\javaw.exe (Nov 17, 2014, 11:0
Calling buildBox with coordinates (25,25) and (50,50):
Box: <25, 25, 50, 50>

Calling buildBox with points (10,10) and (20,20):
Box: <10, 10, 20, 20>

Calling buildBox with point (10,10), width 50 and height 50:
Box: <10, 10, 60, 60>
```

Let’s Examine the Code

Lines 4 – 7: create a new instance of box, initialize instance variables

Lines 9 – 15 – buildBox() instance method sets the variables to their correct values (setter method)

- Since the arguments have the same name as the instance variables, we use keyword this inside of the method to refer to the method’s instance variables

Lines 17-23 & 25-31: Point object is used since x and y values are contained in the object. Overload buildBox() by changing its arguments to create alternate versions of buildBox() – line 1 imports the java.awt.Point class

Lines 33-36: printBox() to display the coordinates

Java Applications

We now have ALL of the needed components to build a full program!

We build an application by assembling one or more classes into a single unit that has no predefined size

main() – the JVM calls this class method when we need to start a program

- Its signature is

```
public static void main(String[ ] arguments) {
        // body
}
```

public – method available to other classes and objects
static – all class methods must be declared as static
void – doesn't return a value
One parameter – an array of strings to hold the command line arguments

Passing arguments to a Java Application

When passing in arguments to run an application, Java stores the arguments in a String array that is passed to the main() method – this array is named in the call to main()

public void static main(**String [] args**) {}

The body of the main() method loops thru args and handling them appropriately

Exercise 1: Averager.java – Page 105

In this exercise we compute a sum and average for a set of numbers

Name this file Averager.java.

We create a class that takes in a set of command-line arguments computes their sum and average.

When ready to run, use the run configurations > arguments tab to enter the values you choose. Be sure to use at least two values

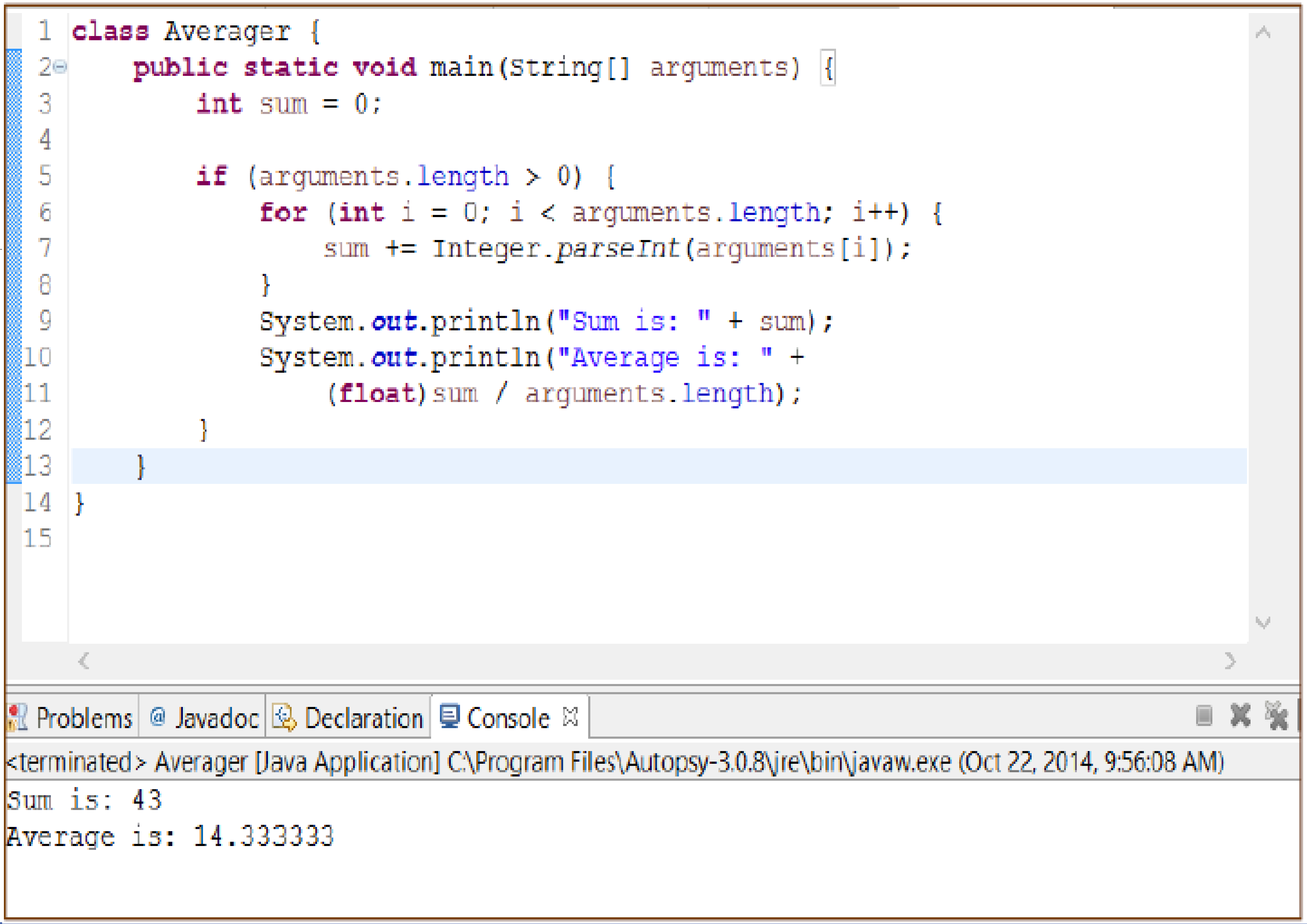
Averager.java

I used 5 14 24 as my input

Line 5: check the length of the arguments array to ensure there is data – if no data, there will be a crash

Lines 6 – 8: iterate thru arguments using a for loop

Line 7 – converts the string value from the array to an int



```
 1 class Averager {
 2     public static void main(String[] arguments) {
 3         int sum = 0;
 4 
 5         if (arguments.length > 0) {
 6             for (int i = 0; i < arguments.length; i++) {
 7                 sum += Integer.parseInt(arguments[i]);
 8             }
 9             System.out.println("Sum is: " + sum);
10             System.out.println("Average is: " +
11                 (float) sum / arguments.length);
12         }
13     }
14 }
15 
```

```
<terminated> Averager [Java Application] C:\Program Files\Autopsy-3.0.8\jre\bin\javaw.exe (Oct 22, 2014, 9:56:08 AM)
Sum is: 43
Average is: 14.333333
```