Over and Over Again

COMP163

“There is no harm in repeating a good thing.”

Plato

ZyBooks Reading

- Read chapter 7 of the ZyBooks textbook
- Answer all of the participation questions in sections 7.1 through 7.9
- Due by midnight on Thursday, October 4

Wireless Emergency Alert

- The Federal Emergency Management Agency (FEMA) will conduct the first-ever test of the Wireless Emergency Alert system around 2:18 today
- Cell phones should buzz and make a loud tone

Looping Structures

- Java provides three different ways to create a loop

  `while` loops

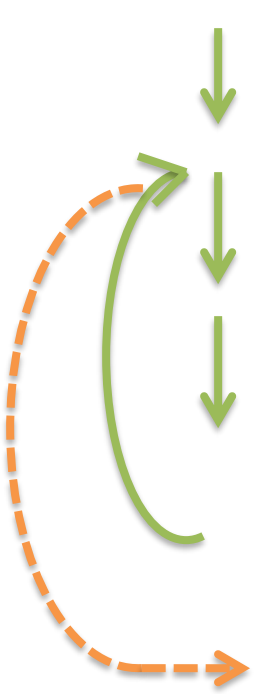
  `do while` loops

  `for` loops

- All of them cause the program to repeat a set of statements

Java **while** statement

- A **while** statement is like an **if** statement that repeats until the logical expression is false



```
int cat = 47, sum = 0;
while (cat > 0) {
    cat = keyboard.nextInt();
    sum = sum + cat;
}
System.out.println( sum );
```

Loop Parts

- A while loop has a loop condition and a body
- The body is repeated while the loop condition is true

```
while (loop condition) {
    // loop body
}
```

Summations in Mathematics

- In mathematics you can specify a sum as

$$sum = \sum_{i=1}^{n} \frac{1}{2i}$$

- which is equivalent to

$$sum = \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \frac{1}{2n}$$

Summations in Java

- We can write the same sum in Java

$$sum = \sum_{i=1}^{n} \frac{1}{2i}$$

```
double sum = 0.0;
int i = 1, n = something;
while ( i <= n ) {
        sum += 1.0 / (2.0 * i);
        i++;
}
```

Calculating Terms

- A more complex summation is

$$e^x = \sum_{i=0}^{\infty} \frac{x^i}{i!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$$

- The factorial of a number **n** is 1*2*3*…*n
- In this summation

$$\text{next term} = \text{previous term} * \frac{x}{i}$$

Sum in Java

```
double x, sum, next, i = 2.0;
// set x to a value
sum = 1.0 + x;     // first two terms
next = x;
while ( next > 0.0001 ) {       // until only small changes
        next = next * x / i;
        sum += next;
        i = i + 1.0;
}
```

What is displayed?

```
int frog = 0;
while ( frog < 3 ) {
    System.out.print(frog);
    frog++;
}
```

A. 0 1 2 3

B. 1 2 3

C. 0 1 2

D. 1 2

E. none of the above

Patterns

- We have learned a few programming patterns that perform common tasks
  - read until a value
  - repeat a fixed number of times
- These patterns are not just one Java statement, but are several Java statements
- These patterns appear in many programs

Priming Read

```
System.out.print("enter number >");
cat = keyboard.nextInt();
while (cat != endValue ) {
        // do something with cat
        System.out.print("enter number >");
        cat = keyboard.nextInt();
}
```

Same

Repeat a Fixed Number of Times

- Starting with one

```
int counter = 1;
while (counter <= maxValue) {
        // do something
        counter++;
}
```

Repeat a Fixed Number of Times

- Starting with zero

```
int counter = 0;
while (counter < maxValue) {
        // do something
        counter++;
}
```

do while loops

- The do while loops is very similar to the while loop except that the loop condition is checked at the end of the loop

```
do {
    cow += 1.0 / (2.0 * bull);
    bull++;
} while ( bull <= horse );
```

do while Syntax

- When a do while loop gets to the end, it checks the logical expression. If true, it repeats the loop

```
do {
    // loop body
} while (logical expression);
```

Always Once

- A do while loop is always executed at least once

```
int hen = 5;
while ( hen < 4 ) {
  hen = hen * hen;  // never executed
}

do {
  hen = hen * hen;  // executed once
} while( hen < 4 );
```

What is displayed?

```
int cow = 10, goat = 3;
while (goat > 0) {
        cow = cow – goat;
        goat--;
}
System.out.println( cow );
```

A. 4
B. 5
C. 7
D. 10
E. None of the above

Not Repeat Until

- When the logical expression of a **`do while`** loop is true, the loop repeats
- Write the logical expression to express when the loop repeats
- Some students incorrectly think a **`do while`** is a repeat until where the logical expression tells you when to stop

Invert Repeat Logic

- In the loop body compute a new term
- Repeat until the term is **less than** 0.001

```
do {
    term = term * x / i;
    sum += term;
    i++;
} while (term >= 0.001);
```

Watch the semicolon

- The following loop is wrong:

```
int cat = 0;
while (cat < 10);
{
      System.out.println("cat is "+cat);
      cat++;
}
```

Logic Error

- With the do loop, the semicolon is required:

```
int cat = 0;
do {
      System.out.println("cat is "+cat);
      cat++;
} while (cat<10);
```

Correct

What is displayed?

```
int cat = 5, dog = 3;
while ( cat < dog ) {
   System.out.print( dog );
   dog++;
}
```

A. 3

B. 3 4

C. 3 4 5

D. nothing

Now what is displayed?

```
int cat = 5, dog = 3;
do {
    System.out.print( dog );
    dog++;
} while ( cat < dog );
```

A. 3

B. 3 4

C. 3 4 5

D. nothing

Reading Files

- Java programs can read input from files as well as from the keyboard
- The Scanner method can be used to read files
- There are also many other classes and methods that can be used to read a file

Scanner with Files

- When you create an object of the Scanner class, you can specify a File object instead of System.in

```
java.io.File  elephant = new java.io.File("mydata.txt");
java.util.Scanner pachyderm =
                              new java.util.Scanner( elephant );

        // read a number from the file
double mammoth = pachyderm.nextDouble();
```

Reading with Scanner

- Reading from a file with a Scanner is just like reading from the keyboard
- All the usual methods are available
- Useful methods are the **`hasNext()`** methods which are true if there is more data in the file

Useful Scanner Methods

- **nextInt**() – read an int
- **nextDouble**() – read a double
- **nextLine**() – read the whole line as a String
- **next**() – read the next word as a String
- **hasNext**() – true if there is more *data*
- **hasNextInt**() – true if there is another int
- **hasNextDouble**() – true if there is another double
- **close**() – close the file when done

File Actions

- Files have to be opened first
  - Scanner and PrintWriter open files
- Data is then read or written to the file
- The file should be closed when the program is done using them
  - If you do not close the file when the program terminates, the system will close it for you

When reading from a file you should

A. Prompt the user before each read
B. Not display any prompts
C. Prompt only for the first read
D. Write prompts to the file

Exceptions

- When a Java program encounters an error during execution it "throws an exception"
- Exceptions can be caused by many, many different errors
- When an exception occurs, the default action is to display an explanation of the error and a "stack trace"

Stack Trace

- A stack trace shows which method called which method

```
Exception in thread "main" java.io.FileNotFoundException:
at java.io.FileInputStream.open(Native Method)
at java.io.FileInputStream.<init>(FileInputStream.java:38)
at java.util.Scanner.<init>(Scanner.java:656)
at ReadFile.main(ReadFile.java:9)
```

- On line 9 of my main method in my ReadFile class, my program called a Scanner method
- The Scanner method called a FileInputStream method
- FileInputStream detected an error in open method

Throwing an Exception

- A method header may include a **throws** clause
- When a method throws an exception, the calling method is responsible for handling it
- When the main method throws an exception, the system will take the default action

```
int methB() throws Exception {
      // do something
}
void methA() {
    int dog = methB(); // must handle exception
}
```

Catching an Exception

- It is possible for a program to catch a thrown exception and do something to recover

```
try {
    Scanner  fileIn = new Scanner(inputFile);
} catch (java.io.FileNotFoundException e) {
    System.out.println("Your data is lost");
}
```

- This is a topic for GEEN165

Scanner File Exceptions

- When you create a Scanner object using a File object, it could throw an exception
- Your program **must** handle the possible exception
- The easiest way to do this is to throw the exception

```
public static void main( String[] unused ) throws
                                    java.io.IOException {
```

```
/*  Example adding the numbers in a file */
public class ReadFile {
   public static void main(String[] dog) throws
                                                    java.io.IOException {
      java.util.Scanner keyboard = new
                                          java.util.Scanner(System.in);
      System.out.print("Enter the filename >");
      String goat = keyboard.next();

      java.io.File  frog = new java.io.File( goat );
      java.util.Scanner  fish = new java.util.Scanner( frog );
      int cow,  sum = 0;
      while ( fish.hasNextInt() ) {
            cow= fish.nextInt();
            sum = sum + cow;
      }
      System.out.println("The sum is " + sum);
      fish.close();
   }
}
```

Exception Classes

- Like almost everything in Java, exceptions are objects
- When your program throws an exception, the system makes an object of the proper exception type
- Exception classes use inheritance
- If a method throws a parent class, it will throw all children of that class

Exception Inheritance Example

- java.lang.Object
  - java.lang.Throwable
    - java.lang.Exception
      - java.io.IOException
        - java.io.FileNotFoundException

- If a method throws IOException, it will also handle FileNotFoundException

Differences between reading from a file and the keyboard include

A. Create the Scanner using a File object not System.in
B. Throw an exception
C. All of the above
D. None of the above

Closing Files

- After using a file, you should close it
- You can call the close() method on a Scanner object
- Closing a file releases the file and any program memory used by the file
- If your program was writing to the file, close will make sure everything is written to disk

Reading from the Web

- In Java you can read a file from a web server in almost the same way you read a file
- You need to create a **java.net.URL** object instead of a java.io.File object
- Instead of passing the File object to Scanner use **`openStream()`** method of a URL object
- You have to throw java.io.IOException

```
/*  Example reading a web file */
public class ReadWeb {
   public static void main(String[] args) throws
                                        java.io.IOException {
      java.util.Scanner keyboard = new
                     java.util.Scanner(System.in);
      System.out.print("Enter the URL >");
      String filename = keyboard.next();

      java.net.URL webFile = new java.net.URL(filename);
      java.util.Scanner  fileIn = new
            java.util.Scanner(webFile.openStream());
      String line;
      while ( fileIn.hasNext() ) {
          line = fileIn.nextLine();
          System.out.println( line );
      }
      fileIn.close();
   }
```

You should close a file

A. After each read
B. After you have read everything from it
C. At the very end of the program
D. Don’t bother, the system will close it

Java Security

- There are some files on your computer that you are not allowed to read or write
- If you attempt to perform a prohibited action, the Java Security Manager will throw a SecurityException
- Java applets are not allowed to read or write files on the computer's disks

Data You Don’t Want

- Sometimes the data file has more data than you want
- If there are five numbers on a line and you only want numbers 2 and 4, it may be easiest to read all five numbers and ignore the ones you don’t need
- The Scanner method nextLine() will read the rest of the line

Skipping Data

- Imagine a data file looks like:

92 3/15/2009 14.5 Fred Smith

47 7/4/1953 3.75 Mary Jones

32 4/1/2013 42.1 Mike J. Snodgrass

- You need the first and third numbers on a line

```
int num1 = infile.nextInt();
infile.next();
double num2 = infile.nextDouble();
infile.nextLine();
```

ZyBooks Reading

- Read chapter 7 of the ZyBooks textbook
- Answer all of the participation questions in sections 7.1 through 7.9
- Due by midnight on Thursday, October 4