Flow of Control:
Repetition

•••

Lecture 3

Dylan Boliske

Loops

- A loop is a repetition control structure.
- It causes a single statement or block of code to be executed while some expression is true.

While Statement

```
while (expression) {

    //loop body

}
```

NOTE: *loop body* can be a null statement, a single statement, or a block of code.

Loop Classifications

- Event-controlled loops : something happens inside the body of the loop that causes the repetition to stop
- Count-controlled loops : the loop repeats a specific number of times

Event-Controlled Loop Types

- Sentinel controlled : keep processing data until a special value, which is not part of the data's value, is entered to indicate that processing should stop
- End-of-file controlled : keep processing data as long as there is more data in the file
- Flag control : keep processing data until the value of the flag changes in the loop body

Sentinel Controlled Loops

- Requires an initial read, meaning that the first data value must be read prior to entering the *while* loop.
- Process the data value(s) and then read in the next value at the end of the loop body.

File Objects

Creating an Instance of File

- A File object is used for representing a file in Java for reading and writing.
- For example:

```
File file = new File("input.txt");

//process file

file.close();
```

Reading Input from a File

- Reading input from a file can be done using a File and a Scanner.

```
File inputFile = new File(“input.txt”);

Scanner input = new Scanner(inputFile);
```

- Reading input from a file will throw an exception, which will be covered in a later lecture. For now, you will need to modify your main method’s head to look like the following:

```
public static void main(String[] args) throws IOException
```

End-of-File Controlled Loops

- Relies on the fact that hasNext returns false if there is no more data available
- For example

```
//open file and scanner

while (input.hasNext()) {

    //read line

    //process data

}

//close file and scanner
```

Flag Controlled Loops

- Use a meaningful name for your flag.
- Initialize the flag (either true or false) before the loop.
- Test the flag in the loop’s test expression.
- Change the value of the flag in the loop body when the correct conditions occur.

Exercise

- Count the sum of the first 10 odd numbers in a data file.
  - Initialize notDone to true
  - Use while(notDone) for the loop test
  - Change the flag to false when 10 odd numbers have been read or if EOF has been reached first.

Nested Loops

- There are some cases where it is necessary to have a loop within another loop. In these cases, the structure should look something like the following:

```
//initialize outer loop

while (outerCondition) {

      …

      //initialize inner loop

      while (innerCondition) {

            //inner loop processing and updating

      }

      …

}
```

Loop Errors and Testing

- Beware of infinite loops - the program does not stop.
- Check loop terminating condition and look out for OBOB (Off By One Bugs).
- Use algorithm walk-through to verify that the appropriate conditions occur in the right places.
- Trace execution of loop by hand with code walk-through.
- Use a debugger (if available) to run the program in “slow motion” or use debugging output statements.

Do Statement

- The do statement is a loop control structure in which the loop condition is tested *after* the body of the loop.

```
do {

    //statement

} while (expression);
```

- The loop body can be a single statement or a block of code.

Do vs While

Do	While
POSTTEST loop (exit condition)	PRETEST loop (entry-condition)
The loop condition is tested after executing the loop body.	The loop condition is tested before executing the loop body.
Loop body is always executed at least once.	Loop body might not be executed at all.

Count-Controlled Loops

- Count-controlled loops contain the following:
  - An initialization of the loop control variable
  - An expression to test for continuing the loop
  - An update to the loop control variable to be executed with each iteration of the body

Count-Controlled Loop Pattern

```
int loopCount = 1;

while (loopCount <= 10) {

    …

    loopCount++;

}
```

Count-Controlled Do Statement

```
int sum = 0;

int counter = 0;

do {

    sum = sum + counter;

    Counter++;

} while (counter <=10 );
```

For Statement

```
for (initialization; expression; update) {
    //loop body
}
```

Example

- What are the results of this loop?

```
int count;

for (count = 1; count <= 10; count++);

    System.out.println(“*”);
```

Output

- No output from the for loop! Why?
- The ; right after the () means that the body statement is a null statement
- In general, the body of the for loop is whatever statement *immediately* follows the ()
- That statement can be a single statement, a compound statement, or a null statement
- Actually, the code outputs one * after the loop completes its counting from 1 to 11