

## SimpleInput.java

- SimpleInput is a simple class to do I/O for you in Java. It is set up to get 1 value/line and doesn't do a lot of error checking.
- I'm not going to worry about what exactly is going on in SimpleInput but on its interface.

## Javadoc

Java has a documentation generator that sifts through the `/**` comments and tries to document your class.

The documentation for SimpleInput was generated this way.

## Using SimpleInput

```
/**
 * main program
 * This main program is only here to show an example of how
 * to use the class. It does nothing other than use each of
 * the methods in the class.
 *
 * @param args command line arguments -- not used in this
 * program
 */
public static void main(String[] args) {
    // TODO Auto-generated method stub
    double dvalue;
    long lvalue;
    String svalue;

    System.out.print("Input a double : ");
    dvalue = SimpleInput.inputDouble();
    System.out.println("Value read was "+dvalue);
}
```

```
System.out.print("Input a long : ");
lvalue = SimpleInput.inputLong();
System.out.println("Value read was " + lvalue);

System.out.print("Input a String : ");
svalue = SimpleInput.inputString();
System.out.println("Value read was " + svalue);

System.exit(0);
}
```

## Loops

September 12, 2006

## Repeating Operations

- In project 2 you could have a method to throw a balloon in your class.
- You need to throw 3 balloons though...
- `throwBalloon( );`  
`throwBalloon( );`  
`throwBalloon( );`
- Other things would be tough though – what if we allowed the player to throw balloons until they finally hit the mascot?

## Three Loops in Java (and C++)

- while
  - Tests at the **top** of the loop
  - The other two types of loops can be rewritten as while loops.
- do-while
  - Tests at the **bottom**
- for
  - Tests at **top** and adds some shortcuts to traditional while loop.

## The while loop

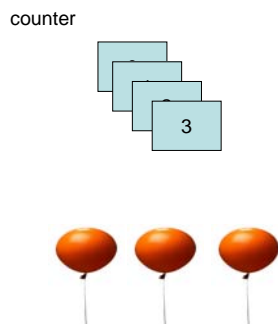
- Syntax:

```
while (conditional) {  
    // loop body – statements to be repeated go here  
}
```

  - The loop is done as long as the conditional is true.
  - If the conditional is originally false the loop is done 0 (zero) times
  - Need to make sure the conditional will eventually become false! (otherwise we get an infinite loop)

## Using it for our 3 tosses

```
BalloonGame game;  
int counter;  
  
counter = 0;  
while (counter < 3) {  
    game.throwBalloon( );  
    counter = counter + 1;  
}
```



## What if we wanted to do it until we got a hit?

- assume we have a method `getHits( )` that returns the number of hits.

```
while ( game.getHits( ) == 0 ) {  
    throwBalloon( );  
}
```

## do-while

- A do-while loop looks somewhat like the while loop.
- Syntax:

```
do {  
    // statements to be repeated  
} while ( condition );
```
- The statements in the loop (or the "body of the loop") are **always** done at least once since we don't test until we reach the end...

## The same examples with a do-while

- Since we would expect either of the two previous examples to do the body of the loop at least once the resulting loops are very easy.

```
count = 0;  
do {  
    game.throwBalloon( );  
    count++; // This is shorthand for  
             // count = count + 1  
} while (count < 3);
```

## The example where we Throw Balloons until we get a hit

```
do {  
    game.throwBalloon( );  
} while (game.getHits( ) == 0);
```

## Common Errors with Loops

- Common errors are:
  - Off by 1
    - either do 1 too many or 1 too few times through the loop.
  - Forget to change the variable being tested in the loop's conditional
    - The cause of more infinite loops...
  - Forget to initialize the variable being tested in the loop's conditional

## The for Loop

- The for loop is basically a while loop in disguise. It is set up to take care of a certain common class of loops.
- Syntax:

```
for ( initializer; conditional; iterator ) {  
    // Body of the loop  
}
```

- The initializer is a statement done before the loop is entered.
- The conditional is the test used for the loop. The loop is done as long as the test is true and the test is performed at the top of the loop.
- The iterator is a statement done after the body of the loop is done and before we go back to the top of the loop and test the conditional.

## Example for 3 Tosses

```
for ( count = 0; count < 3; count = count + 1 ) {  
    game.throwBalloon( );  
}
```

This tends to bring out the "counting" variable and places all 3 things that affect it in the same line.

Can be used for things other than counting too but that is its most common use.

## How does that relate to a While Loop.

```
for (initializer; conditional; iterator) {  
    // Body of loop  
}
```

Becomes:

```
initializer;  
while (conditional) {  
    // Body of loop  
    iterator;  
}
```