Variable Scoping, Input, and Output

Variable Scoping

```java
public class CatTest
    extends TestCase
{
    private Cat testCat;

    public void setUp()
    {
        testCat = new Cat();
    }
}
```

Scope

- method's parameters have local scope

Initializing review

```java
private int value;
    public Answer(int v)
    {
        this.value = v;
    }
```

Basic I/O

Input operations are framed in terms of reading from a stream in a three-step process:

1. open the stream
2. read data items from the stream front to back in sequence
3. close the stream.

Output operations are framed in terms of writing to a stream in a similar three-step process:

1. open the stream
2. write data onto the end of the stream in sequence
3. close the stream.

```java
import java.io.*;
import java.util.*;
import student.*;
```

Output using PrintWriters

```java
PrintWriter outStream = IOHelper.createPrintWriter("output.txt");
outStream.print("This is a message, and ");
```

```
outStream.println("these words appear on the same line as those above"); // next line will be o
n a different line
outStream.println(100 / 2);  // prints the value "50"
outStream.write(65);         // writes the letter 'A', whose ASCII code is 65
outStream.close();

// if you want to split output over multiple lines call the print writer as the parameter
public void printHeader(PrintWriter outStream)
{
    outStream.println("This is the output for ...");
    // other output commands go here.
}

// default output
System.out.println("beginning the code ...");
```

Input using Scanners

```java
Scanner inStream = IOHelper.createScanner("input.txt");
// scanner methods
<scanner>.hasNext();
<scanner>.next();
<scanner>.hasNextLine();
<scanner>.nextLine();
<scanner>.hasNext<PrimitiveType>(); // .hasNextInt , hasNextDouble, etc
<scanner>.next<PrimitiveType>();
<scanner>.useDelimiter(String pattern); // set whitespace symbols to change breaking up inp
ut
<scanner>.close();

if (inStream.hasNextLine())
    String thisLine = inStream.nextLine(); // moves to the line
    Scanner line = new Scanner(thisLine); // Scans the line
    System.out.println(line.nextInt());

// other scanner inputs
Scanner keyBoard = IOHelper.createKeyboardScanner();
String name = keyBoard.nextLine();

Scanner inWebFile = IOHelper.createScannerForURL(
    "http://server.subdomain.domain/dir/file.txt");
String line = inWebFile.nextLine();

// spread input across methods (same as output, use scanner as param)
public void readHeader(Scanner inStream)
{
    String nextLine = null;
    if (inStream.hasNextLine())
    {
        nextLine = inStream.nextLine();
        // other input commands go here.
```

Testing

```java
// test class MUST extend TestCase
// input
public void testProcessSomeInput()
{
    // set up the input stream
    setIn("some test input");

    // run the method to get results
    doIt.processSomeInput(in());

    // test that the result is what was expected
    assertThat( ... );
}

out().getHistory();
// check what is inside PrintWriter

// output
public void testProcessSomeInput()
{
    // run the method to get results
    doIt.produceOutput(out());

    // test that the result is what was expected
    assertThat("what I expect").isEqualTo(out().getHistory());
}
```