Recitation 2 - January 16th A4/GR

William and Isaac

Homework Questions/Checkstyle

- Checkstyle is a program we use to encourage good code formatting
- It is a JAR file which you can download at http://cs1331.gatech.edu/cs1331-style-guide.html
- Biggest rules
  - Use 4 spaces instead of tabs
  - Class names are UpperCamelCase, variables are lowerCamelCase
  - No lines longer than 80 characters; break lines after commas or before operators
  - Curly braces go on the same line as the class declaration/if/else/while/for/etc.

Running Checkstyle

- To run checkstyle, run the following
  - `java -jar checkstyle-6.2.2.jar *.java`
- Just javadocs
  - `java -jar checkstyle-6.2.2.jar -j *.java`
- Checkstyle + javadocs
  - `java -jar checkstyle-6.2.2.jar -a *.java`
- To run checkstyle on a single file, you can run
  - `java -jar checkstyle-6.2.2.jar <file name>.java`

Variables

Variable Creation

- To **declare** a variable, we do:
  - `<type> <variable name>;`
- To **assign** a value to a variable, we do:
  - `<variable name> = <value>;`
  - When we assign a value to a variable for the first time, we call it **initializing**
- We can do this all in one line if we want!
  - `<type> <variable name> = <value>;`

Example

```
1  public class Test {
2      public static void main(String[] args) {
3          // Declaration
4          long bookISBN;
5          String bookTitle;
6
7          // Assignment
8          bookISBN = 9783161484100;
9          bookTitle = "The Name of The Wind";
10
11         String bookGenre = "Fantasy";
12     }
13 }
```

Variable Naming

- Good variable naming is essentially for readability
- Be descriptive
  - Use `parkingSpace` instead of `i`
  - Use `sum` instead of `x`
  - `z = x / y;` is hard to read versus `average = sum / numElements;`
- Java uses the camelCase convention
  - For names with multiple words, capitalize the first letter of each word (except first)
- Can't use reserved words
  - `public, static, class, int, double, float`

What is Wrong?

```
1   public class Test {
2       public static void main(String[] args) {
3           int NumberOfPies = 27;
4           System.out.println("We have " + NumberOfPies + " Pies");
5       }
6   }
7
```

```
/Users/williamcheng/Desktop/CheckStyleExample/Test.java:3:13:
Name 'NumberOfPies' must match pattern '^[a-z][a-zA-Z0-9]*$'.
Audit done. Errors (potential points off):
1
```

What does a variable hold?

Primitives

When a variable is a primitive type, the variable stores its value

- byte (8 bits)
- short (16 bits)
- int (32 bits)
- long (64 bits)
- **float** (32 bits)
- **double** (64 bits)

bold = decimal numbers

- char (16 bits)
- boolean (8 bits?)

Narrowing vs Widening

- Narrowing:
  - Bigger goes to smaller (loses precision)
  - `double` -> `int`
- Widening
  - Smaller goes to bigger (doesn't lose precision)
  - `byte` -> `int`

Conversion

- In Java, there are three main ways to convert between primitives
  - Assignment (implicit casting)
    - `double sum = 5;`
  - Arithmetic promotion (implicit casting)
    - `double quotient = 5.0 / 2;` or `double x = 2 + 2.0;`
  - Casting explicitly (explicit casting)
    - `int badNum = (int) 3.14;`

Operators

Category	Operator	Name/Description	Example	Result
Arithmetic	+	Addition	3+2	5
	-	Subtraction	3-2	1
	*	Multiplication	3*2	6
	/	Division	10/5	2
	%	Modulus	10%5	0
	++	Increment and then return value	X=3; ++X	4
		Return value and then increment	X=3; X++	3
	--	Decrement and then return value	X=3; --X	2
		Return value and then decrement	X=3; X--	3
Logical	&&	Logical “and” evaluates to true when both operands are true	3>2 && 5>3	False
	||	Logical “or” evaluates to true when either operand is true	3>1 || 2>5	True
	!	Logical “not” evaluates to true if the operand is false	3!=2	True
Comparison	==	Equal	5==9	False
	!=	Not equal	6!=4	True
	<	Less than	3<2	False
	<=	Less than or equal	5<=2	False
	>	Greater than	4>3	True
	>=	Greater than or equal	4>=4	True
String	+	Concatenation(join two strings together)	“A”+”BC”	ABC

Order of Precedence

Order of Precedence	Operators	Description
1	(unary negation) !	Unary negation, logical NOT
2	* / %	Multiplication, Division, Modulus
3	+ -	Addition, Subtraction
4	< > <= >=	Less-than, Greater-than, Less-than or equal to, Greater-than or equal to
5	== !=	Is equal to, Is not equal to
6	&&	Logical AND
7	||	Logical NOT
8	= += -= *= /= %=	Assignment and combined assignment operators.

Objects

When a variable is an object/reference-type, it stores a reference to the object

- String
- Scanner
- A lot, lot more…

Instantiation is creating a new instance of an object

Strings (Our First Objects!)

- Strings are special
  - `String s = "Hello, world!";`
  - `String s = new String("Hello, world!");`
- Strings are immutable
  - Concatenating Strings doesn't mutate them, it creates new ones
- You can find many, many methods in the API