Review Ch 1-3
Executing code

Compile code
(convert from Java to computer code)
- Syntax errors will prevent compilation

Run code
- Runtime errors will crash your program
- Logic errors will make your program give the wrong output
Identifiers

The identifier is the name of a variable/method
- Case sensitive
- Must use only letters, numbers or _
- Cannot start with a number
- (Some reserved identifiers)

Examples (second word):
    int x, String s_o_s, double high2low
Primitive Types

**boolean** - True or false
**char** - (character) A letter or number
**int** - (integer) Whole numbers
**long** - (long integers) Larger whole numbers
**float** - Decimal numbers
**double** - Larger decimal numbers

**doubles** are approximations
**ints** are exact but have a more limited range
Operations

Order of precedence (higher operations first):
-\-, +, ++, -- and ! (unary operators)
*, / and % (binary operators)
+ and - (binary operators)

Operators that change variables:
++, --, +=, -=, *=, /=, =

Note: integer division happens if you divide two ints: int / int = int
Java Strings

String is an immutable class

You can concatenate a String with anything else using a + (same precedence as addition)

Strings contain a sequence of **chars**, each labeled by an index

```
Hello
0 1 2 3 4
```
Strings methods

Useful String methods:
length() = how many letters are in the string
indexOf() = find letters or words in a string
substring() = pull out a chunk of letters from the string

More random ones:
trim() = remove white space at the end
toUpperCase() = CHANGE WORD TO CAPS
To create a Scanner object:
```java
Scanner x = new Scanner(System.in);
```

Scanner methods:
- `nextInt()` - reads the next integer
- `nextDouble()` - reads the next double
- `next()` - reads the next word
- `nextLine()` - reads until you hit return (newline character)
If statements

```java
if (boolean expression) {
    // code
}
else {
    // more code
}
```

 Logical operations:

- `>` (greater than)
- `==` (equals)
- `<` (less than)
- `>=` (greater than or equal to)
- `!=` (not equal to)
- `<=` (less than or equal to)

|| is the OR operations

&& is the AND operations
Short-circuit evaluation

Simple cases of short-circuit:

When you have a bunch of ORs

if( expression || exp || exp || exp )

Once it finds any true expression, if statement will be true

When you have a bunch of ANDs

if( expression && exp && exp && exp )

Once it finds any false expression, if statement will be false
Scope

Variables only exist in the most recently started block:

```java
if (x < y)
{
    int z = 9;
}
```

- z lives in most recent block
- z goes away at corresponding closing block

If you want variables to exist longer, you need to declare them further up in the program.
Loops

3 parts to any (good) loop:
- Loop variable initialized
- boolean expression with loop variable
- Loop variable updated inside loop

for loops have these 3 parts in the same place
while loops have these spread out
do while loops are while loops that always execute at least once
Looping control commands

**continue** restarts loop immediately

```latex
\begin{verbatim}
for (i = 0; i < 10; i++)
{
    // code will run everytime
    if (doSkip)
    {
        continue;
    }
    // code will not run
    // if doSkip is true
}
\end{verbatim}
```

**break** stops loop

```latex
\begin{verbatim}
for (i = 0; i < 10; i++)
{
    // code
    if (doSkip)
    {
        break;
    }
    // outside loop code
\end{verbatim}
```