Set 1: Variables

Skill 1.1: Declare and initialize a variable

Skill 1.1 Concepts

Variables are letters or words that can be used to store data. Variables declared inside of a method are referred to as *local* variables - that is their use is localized to the method in which they are declared.

Java is a strict variable type language. That is, to declare a variable requires that you not only provide a name for the variable, but also declare the type. To illustrate how to declare a variable in Java, we will use the *int* type variable.

An int type variable can hold any whole number. The following code could be used to declare an *int* type variable called *x*. Notice that in the code, *int* is the type of variable we want to declare and *x* is the name of the variable.

```
int x;
```

The variable we declared above is currently empty, that is no value has been assigned to it. Assigning a value to a variable is called *initializing*. The following code illustrates how to initialize the variable x. Notice in the code below,

- We did not type *int* again. Once a variable is declared, it cannot be declared again. Leaving *int* in front of our variable x would actually generate an error.
- Also, notice that the variable name is always on the left of the equal sign. The value you want to assign to the variable goes on the right of the equal sign.

```
x = 10;
```

The value of variables can be changed at any point in the method in which they were declared. For example, the code below, would change the value of x to 5.

```
x = 5;
```

Variables can also be manipulated. For example I could add 1 to variable x as follows,

```
x = x + 1;
```

The naming convention for variables in java is "lower camel case". That is, the first letter in the variable is lower case. If there are multiple words associated with the variable name, they are run together. Each word in the variable (except for

the first word) is capitalized. Consider the example below. In this example, we have declared and initialized the variable on one line.

```
int myFinalScore = 10;
```

To print the value of a variable to the console, we use the same print statement we learned previously. The following code could be used to print *myFinalScore*. Notice in the below example, there are no quotes around the variable. If there were quotes around myFinalScore, "myFinalScore" is what would be printed. Leaving the quotes off, prints the value of the variable.

```
System.out.prinlnt(myFinalScore);
```

Skill 1.1 Exercise 1

Skill 1.2: Describe the 8 primitive data types

Skill 1.2 Concepts

There are eight primitive variable types in the Java language. These variable types are the most basic data types available within the Java language and serve as the building blocks of data manipulation. Primitive variable types serve only one purpose — to contain pure, simple values of a kind. Because these data types are defined into the Java type system by default, they come with a number of operations predefined.

Primitive number type variables are divided into two groups:

Integer types stores whole numbers, positive or negative (such as 123 or -456), without decimals. Valid types are byte, short, int and long. Which type you should use, depends on the numeric value.

The four types of Integer types are described below. Each of the Integer types store whole numbers. The only difference is the range of numbers they can store.

Data Type	Size	Description
byte	1 byte	Stores whole numbers from -128 to 127
short	2 bytes	Stores whole numbers from -32,768 to 32,767
int	4 bytes	Stores whole numbers from -2,147,483,648 to 2,147,483,647
long	8 bytes	Stores whole numbers from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Floating point types represents numbers with a fractional part, containing one or more decimals. There are two types: float and double.

The two types of Floating type variables are described below.

Data Type	Size	Description
float	4 bytes	Stores fractional numbers. Sufficient for storing 6 to 7 decimal digits
double	8 bytes	Stores fractional numbers. Sufficient for storing 15 decimal digits

The last two primitive data types do not store numbers. They are described below,

Data Type	Size	Description
boolean	1 bit	Stores true or false values
char	2 bytes	Stores a single character/letter or ASCII values

For the purposes of this class you only need to be concerned with the following primitive variable types

- int
- double
- boolean
- char (Although not required of AP, we will utilize this in future projects)

Skill 1.2 Exercise 1

Skill 1.3: Declare and initialize the primitive variable types you must know for this class

Skill 1.3 Concepts

As aforementioned, the three variable types you MUST know for the AP Exam are int, double, and boolean. But, because we use the char variable type in upcoming projects, you should be familiar with the use of this data type too. The code segment below illustrates how to declare and initialize each of these variable types,

```
int aWholeNumber = 3;
double aNumberWithADecimal = 3.0;
boolean aTrueFalseVariable = true;
char aCharValue = 'a';
```

- Notice that all the variable declarations above begin with a type declaration followed by an equal sign
- The value of the variable appears to the right of the equal sign.
- The *char* variable type requires single quotes around the value.

Skill 1.3 Exercise 1

Skill 1.4: Interchange int and double variable types

Skill 1.4 Concepts

Recall that an *int* variable type is a whole number. Whereas a *double* is a decimal type number
When a value is assigned to a double type variable, Java automatically assigns zeros after the digit. This makes this type of variable more precise (1.000 for example, is more precise than just 1). Because a double is more precise than an int, you cannot assign a double to an int, however the reverse is allowed. Consider the following example,

```
int i = 1;
double d = 2;
d = i; //allowed because i is an int and is not precise
i = d; //not allowed because d is a double and cannot be assigned to an int
i = 2.0000; //not allowed because 2.0000 is too precise
```

The reason for the above assignment rules is that while asigning an int to a double does not cause a loss in precision or information, assigning a double to an int does. Likewise you may not assign a decimal precision number to an int type variable. If you try this, you will get the following error,

```
error: incompatible types: possible lossy conversion from double to int
int i = 2.000;
1 error
```

You can however force a *double* type variable into an *int* type variable with casting. The following code snippet illustrates how to cast a *double* type variable into an *int*,

```
int i = 1;
double d = 2.9999;
d = i; //allowed because i is an int and is not precise
i = (int)d; //allowed because we casted variable d to an int
i = (int)2.0000; //allowed because we casted 2.0000 to an int
```

Be careful with the above operation however, because int type variables cut off all information following the decimal and only preserve the "ones" place. Converting 2.9999 to an int for example, would result in 2. Rounding does NOT apply to int variables - all numbers after the decimal are simply chopped off.

Skill 1.4 Exercise 1

Skill 1.5: Declare and initialize a Boolean variable type

Skill 1.5 Concepts

boolean variable types can only be assigned a value of true or false. The following code is an example. And, like other variable types, their value can be changed at anytime,

```
boolean b = true;
b = false;
```

Skill 1.6: Declare and initialize a String variable type

Skill 1.6 Concepts

String variable types are not considered primitive variables types. But, because of their utility we will introduce them here. In future lessons we will explore *String* variables in more depth. String variables are used to store things in quotes... like "Hello World!". The following is an example,

```
String s = "My name is Fred";
s = "Now it is Ted";
```

Skills 1.5 & 1.6 Exercise 1

Skill 1.7: Identify legal and illegal variable assignments

Skill 1.7 Concepts

Rules for naming variables

When naming variables, regardless of the type, the following rules apply,

- Variable names must begin with a letter (or an underscore character)
- Variable names cannot contain spaces
- The only "punctuation" character permissible inside a variable name is an underscore "_".
- Variable names cannot be one of the reserved words that are part of the Java library

Legal and illegal variable name examples

Legal	Illegal
Agro	139
D	139Abc
D31	Fast one
Hoppergee	class
hopper_gee	slow.sally
largeArea	Double
computerScience	computer:Science
codeIsCool	code-is-cool

Skill 1.7 Exercise 1