AP COMPUTER SCIENCE
JAVA CONCEPTS II: VARIABLES AND DATA TYPES

PAUL L. BAILEY

1. Variables

A *variable* is a name which exists in the Java code for a piece of data which is stored in the computer's memory. Each variable has a *kind*, which refers how it was created, and a *data type*, which refers to what it stores.

1.1. Variable Names.

1.1.1. *Rules.* Variable names are case sensitive. Variable names may not contain white space. A variable name may be any valid Java identifier:

- Each identifier must have at least one character.
- The first character must be a letter, and underscore, or a dollar sign. The first character can not be a digit.
- The rest of the characters (besides the first) can be a letter, a digits, and underscore, or a dollar sign.
- The variable cannot be a reserved word (for example, `class`, `double`, or `if`).

1.1.2. *Conventions.* The convention, however, is to always begin your variable names with a lowercase letter, not "`$`" or "`_`". Additionally, the dollar sign character, by convention, is never used at all. You may find some situations where auto-generated names will contain the dollar sign, but your variable names should always avoid using it. A similar convention exists for the underscore character; while it's technically legal to begin your variable's name with underscore, this practice is discouraged.

When choosing a name for your variables, use full words instead of cryptic abbreviations. Doing so will make your code easier to read and understand. In many cases it will also make your code self-documenting; fields named cadence, speed, and gear, for example, are much more intuitive than abbreviated versions, such as s, c, and g. Also keep in mind that the name you choose must not be a keyword or reserved word. If the name you choose consists of only one word, spell that word in all lowercase letters. If it consists of more than one word, capitalize the first letter of each subsequent word. The names gearRatio and currentGear are prime examples of this convention. If your variable stores a constant value, such as `static final int NUM_GEARS = 6`, the convention changes slightly, capitalizing every letter and separating subsequent words with the underscore character. By convention, the underscore character is never used elsewhere.

Date: August 20, 2019.

1.2. **Variable Kinds.** The variable kind indicates how the variable was created, and where it may be used.

- *Static* variables are declared inside a class but outside of a method, and exist at the class level. They do not require an instance of an object to be used. They are global, in the sense that they can be accessed by any portion of the program that has access to them. Static variables are also called *class variables*.
- *Instance* variables are declared inside a class but outside of a method, but exist at the object level. They require an instance of an object to be used; moreover, each object of the same class has different values for its instance variables.
- *Local* variables are declared inside a block of code. They are accessible only from inside of the code block in which they are declared.
- *Parameters* are passed into a method as part of the method signature. Otherwise, they behave like local variables.

2. Data Types

2.1. **Java Variable Categories.** Java is a strongly typed language, which means that every variable has a type. There are two categories of types: those which are embedded in the language itself, and those which are defined by classes. The types which are embedded in the language are called *primitive types*. Instances of the types defined by classes are called objects. Primitive types are not objects in Java.

When a variable of a primitive type is passed to a method as a parameter, the value is copied, and the method receives a separate copy of the value. Thus the method cannot change the original value. Thus primitive types are sometimes called *value types*.

When a variable of a class type is passed to a method as a parameter, only a reference to the object is copied; thus the method has the capability of changing the object, if the object allows itself to be changed. Thus class types are sometimes called *reference types*.

2.2. **Java Primitive Types.** Java supports eight different primitive types, which are listed here. Only three of these are tested on the AP Computer Science examination, and those are marked in the table.

Type	AP	Bit Length	Values	Description
`byte`		8	$\pm 1.27 \times 10^2$	8-bit signed integer
`short`		16	$\pm 3.28 \times 10^5$	16-bit signed integer
`int`	$\surd$	32	$\pm 2.14 \times 10^9$	32-bit signed integer
`long`		64	$\pm 9.22 \times 10^{18}$	64-bit signed integer
`float`		32	$\pm 3.40 \times 10^{38}$	32-bit floating point
`double`	$\surd$	64	$\pm 1.80 \times 10^{308}$	64-bit floating point
`boolean`	$\surd$	8	`true` or `false`	boolean
`char`		16	characters	16-bit unicode

- Java does not support unsigned integers.
- 32-bit floating point has approximately 6 digits of precision.
- 64-bit floating point has approximately 15 digits of precision.
- The actual storage space consumed by boolean is machine independent.

2.3. **Java Standard Classes.** Arrays in Java are a special type of object, which are supported within the language itself.

Java comes with a standard library containing hundreds of classes. The nonstatic library classes required for the AP examination are listed here.

- `Object`: The base class for every other Java class. Every Java class extends `Object`.
- `Integer`: A wrapper class for the primitive type `int`.
- `Double`: A wrapper class for the primitive type `double`.
- `Boolean`: A wrapper class for the primitive type `boolean`.
- `String`: A string of unicode characters.
- `List`: A list of objects.
- `ArrayList`: An extension of `List` to mimic or otherwise work with arrays.

The Java programming language provides special support for character strings via the `java.lang.String` class. Enclosing your character string within double quotes will automatically create a new String object; for example, `String s = "this is a string";`. String objects are immutable, which means that once created, their values cannot be changed. The String class is not technically a primitive data type, but considering the special support given to it by the language, you'll probably tend to think of it as such.

2.4. **Default Values.** It's not always necessary to assign a value when a field is declared. Fields that are declared but not initialized will be set to a reasonable default by the compiler. This default will be zero or null, depending on the data type. Relying on such default values, however, is generally considered bad programming style.

The following chart summarizes the default values for the primitive data types.

Data Type	Default Value
byte	`0`
short	`0`
int	`0`
long	`0L`
float	`0.0f`
double	`0.0d`
char	`\u0000`
Object	`null`
boolean	`false`

3. Literals

You may have noticed that the new keyword isn't used when initializing a variable of a primitive type. Primitive types are special data types built into the language; they are not objects created from a class. A literal is the source code representation of a fixed value; literals are represented directly in your code without requiring computation. It is possible to assign a literal to a variable of a primitive type.

3.1. **Integer Literals.** An integer literal is of type long if it ends with the letter L or l; otherwise it is of type int. It is recommended that you use the upper case letter L because the lower case letter l is hard to distinguish from the digit 1.

Values of the integral types byte, short, int, and long can be created from int literals. Values of type long that exceed the range of int can be created from long literals. Integer literals can be expressed in decimal, hexadecimal, or binary:

```
int decVal = 26;        // The number 26, in decimal
int hexVal = 0x1a;      // The number 26, in hexadecimal
int binVal = 0b11010;   // The number 26, in binary
long lngVal = 123456L;  // A long literal
```

3.2. **Floating Point Literals.** A floating-point literal is of type `float` if it ends with the letter F or f; otherwise its type is `double` and it can optionally end with the letter D or d.

The floating point types (float and double) can also be expressed using E or e (for scientific notation), F or f (32-bit float literal) and D or d (64-bit double literal; this is the default and by convention is omitted):

```
double d1 = 123.4;      // Double literal
double d2 = 1.234e2;    // Scientific notation literal
float f1  = 123.4f;     // Floating point literal
```

Additionally, floating point types support three constants `POSITIVE_INFINITY`, `NEGATIVE_INFINITY`, and `NaN` (not a number).

3.3. **Boolean Literals.** The boolean literals are `true` and `false`.

3.4. **Character and String Literals.** Use single quotes for char literals and double quotes for String literals. The Java programming language also supports a few special escape sequences for char and String literals:

Escape Character	Name	ASCII Value
`\b`	backspace	8
`\t`	tab	9
`\n`	line feed	10
`\r`	carriage return	13
`\"`	double quote	34
`\'`	single quote	39
`\\`	backslash	92

3.5. **Object Literals.** There is a special `null` literal that can be used as a value for any reference type. The value `null` may be assigned to any variable, except variables of primitive types.

There is also a special kind of literal called a class literal, formed by taking a type name and appending ".class"; for example, String.class. This refers to the object (of type Class) that represents the type itself.

Reference: `http://docs.oracle.com/javase/tutorial/java/nutsandbolts/`
Reference: `http://www.cs.umd.edu/~clin/MoreJava/Intro/varident.html`

Department of Mathematics, BASIS Scottsdale
Email address: `paul.bailey@basised.com`