AP Computer Science　　　　　　　　　　Name: ______________________________
Chapter 6 Notes (2)

DATA TYPES, VARIABLES, and ARTHMETIC

Numeric Types:

Every data type has a range of values. The compiler allocates memory space for each variable or constant according to its data type. Java provides 8 primitive data types for numeric values, characters, and Boolean values.

Primitive Data Types:

In Java, values of primitive data types are not objects. This is a concession to earlier programming languages, like C, from which Java borrows much of its syntax. Objects have data types, too: the data type of an object is its class.

Java has the following eight primitive data types, designated by reserved words:

Type	Size (bytes)	Range
`boolean`	1	`true` or `false`
`char`	2	Unicode character set (with ASCII subset)
`byte`	1	From $-2^7 = -128$ to $2^7 - 1 = 127$
`short`	2	From $-2^{15} = -32,768$ to $2^{15} - 1 = 32,767$
`int`	4	From -2^{31} to $2^{63} - 1$
`long`	8	From -2^{63} to $2^{63} - 1$
`float`	4	Approx. from -3.4×10^{38} to 3.4×10^{38}
`double`	8	Approx. from -1.8×10^{308} to 1.8×10^{308}

Note that like other Java reserved words, the names of the primitive data types are spelled in lowercase letters. They are called *primitive* because variables of these types do not have the properties of objects (in particular, they do not have any methods.)

Integer Types:　　`byte`, `short`, `int`, and `long`

Floating-Decimal Types:　　`float` and `double`

Because variables of different types occupy different numbers of bytes in memory, we say they have different *sizes*.

Although `float` and `double` can represent a huge range of numbers, their precision is limited to only about seven significant digits for the `float` and twice as many for the `double`.

It is a programmer's responsibility to make sure the values of variables and all the intermediate and final results in arithmetic expressions fit within the range of the chosen data types, and that these types satisfy the precision requirements for computations.

Example 1:

a. What is the value of `value`?

```
int value = 2147483647 + 1;
```

b. What is the value of `value`?

```
int value = –2147483648 – 1;
```

Overflow: When a variable is assigned a value that is too large (in size) to be stored.

Underflow: When a floating-point number is too small (i.e., too close to zero) to be stored.

Numeric Operators:

The operators for numeric data types include the standard arithmetic operators: addition (+), subtraction (-), multiplication (*), division (/), and remainder (%), as shown in the table below. The *operands* are the values operated by an operator.

Name	*Meaning*	*Example*	*Result*
`+`	Addition	`34 + 1`	`35`
`–`	Subtraction	`34.0 – 0.1`	`33.9`
`*`	Multiplication	`300 * 30`	`9000`
`/`	Division	`1.0/2.0`	`0.5`
`%`	Remainder	`20 % 3`	`2`

Example 2: Integer Division

What is printed out in each example?

a.
```
int x = 19 / 5;
System.out.println(x);
```

b.
```
int y = 19 % 4;
System.out.println(y);
```

Example 3:

Suppose you and your friends are going to meet in 20 days. What day is it in 20 days, if today is Saturday?

Example 4:

Write a program that asks a user for an amount of time in seconds and then converts it into minutes and seconds.

Example 5:

What is the output for the following statements? Why?

a.
```
System.out.println(1.0 - 0.1 - 0.1 - 0.1 - 0.1 - 0.1);
```

b.
```
System.outprintln(1.0 - 0.9);
```

Exponent Operators:

The `Math.pow(a, b)` method can be used to compute a^b. The `pow` method is defined in the `Math` class in the Java API. You invoke the method using the syntax `Math.pow(a, b)` (`Math.pow(2, 3)`), which returns the results a^b (2^3). Here `a` and `b` are parameters for the `pow` method and the numbers 2 and 3 are actual values used to invoke the method. For example,

```
System.out.println(Math.pow(2, 3));     // Displays 8.0
System.out.println(Math.pow(4, 0.5));   // Displays 2.0
System.out.println(Math.pow(2.5, 2));   // Displays 6.25
System.out.println(Math.pow(2.5, -2));  // Displays 0.16
```

Numeric Literals:

A literal is a constant value that appears directly in a program.

For example, 34 and 0.305 are literals in the following statements:

```
int numberOfYears = 34;
double weight = 0.305;
```

Integer Literals:

An integer literal can be assigned to an integer variable as long as it can fit into the variable.
A compile error will occur if the literal is too large for the variable to hold.
The statement `byte b = 128`, for example, will cause a compile error, because 128 cannot be stored in a variable of the `byte` type. (Note that the range for a byte value is from -128 to 127.)

Floating-Point Literals:

Floating-point literals are written with a decimal point. By default, a floating-point literal is treated as a double type value. For example, 5.0 is considered a double value, not a float value.

Scientific Notation:

Floating-point literals can be written in scientific notation in the form of $a \times 10^b$. For example, the scientific notation for 123.456 is 1.23456×10^2 and for 0.0123456 is 1.23456×10^{-2}. A special syntax is used to write scientific notation numbers. For example, 1.23456×10^2 is written as `1.23456E2` or `1.23456E+2` and 1.23456×10^{-2} as `1.23456E-2`. `E` (or `e`) represents an exponent and can be in either lowercase or uppercase.