Chapter 6

Primitive types

Lesson page 6-1. Primitive types

Question 1. There are an infinite number of integers, so it would be too ineffient to have a type *integer* that would contain all of them.

Question 2. The five integral types are: `byte`, `short`, `int`, `long`, and `char`.

Question 3. The two floating-point types are `float` and `double`.

Question 4. Type `long` is the integral type with the largest range. It is also the integral type that uses the most amount of memory.

Question 5. Type `double` is the floating-point type with the largest range. It is also the floating-point type that uses the most amount of memory.

Question 6. Type `boolean` has only two values: `true` and `false`.

Question 7. Here's the filled-in table:

type	**smallest value**	**largest value**
boolean	`false/true`	`false/true`
long	$-(2^{63})$	$2^{63}-1$
int	$-(2^{31})$	$2^{31}-1$
short	$-(2^{15})$	$2^{15}-1$
byte	$-(2^{7})$	$2^{7}-1$

Lesson page 6-2. The integral types

Activity 6-2-1: Integral constants (literals)

Question 1. A literal is a Java denotion for a value of a type. For example, `123` is a literal of type `long`.

Activity 6-2-2: Operations on type `int`

Question 2. The summary of operations on type `int` is in a footnote on the lesson page.

	Symbol	**Name of operation**	**Example**
unary operator:	−	negation	`-45`
unary operator:	+	addition	`+45 = 45`
binary operator:	+	addition	`4 + 4 = 8`
binary operator:	−	subtraction	`6 - 4 = 2`
binary operator:	∗	multiplication	`3 * 2 = 6`
binary operator:	/	division	`5 / 2 = 2`
binary operator:	%	remainder	`5 % 2 = 1`

Question 3. False. Java does not stop execution if integers get too large and cause overflow. Instead, the program probably produces incorrect results.

Question 4. The expression `-2147483648-1` evaluates to `2147483647`.

Question 5. We have: `5-(6/7) = 5-(0) = 5`.

Question 6. In Java, division `6/7` results in an integer, with the fraction part being deleted. In mathematics, division of two integers yields a rational number.

Question 7. There are no operations on type `byte` or type `short`.

Question 8. True. All values of one integral type are contained in the integral values of a wider type.

Question 9. True. Java automatically promotes the narrower type `byte` to the wider type `int`.

Question 10. The four integral types mentioned in this activity, from narrowest to widest, are: **byte**, **short**, **int**, **long**.

Activity 6-2-4: Casting integer values

Question 11. To cast a value is to convert it from one type to another.

Question 12. The expression is: `(byte)(32768 - 32765)`.

Question 13. Here are the answers:

```
byte b= 3.5; // can't assign a decimal to a byte
short s= b;

short j= 4;
long l= (long)j;

byte k= 440; // too large to store in a byte
int i= k;
```

Activity 6-2-5: Review of integer types

Lesson page 6-3. A minimalist view of floating-point

Activity 6-3-1: Literals of type double

Question 1. All the literals are of type `double` except the last, `4500`, which is of type `int`.

Activity 6-3-2: Values of type double and operations on them

Question 2. In Java, write 108.225×10^5 as `108.225 * 10e5`.

Question 3. The underlined items are, in order: `double`, `double`, `float`, `float`, `long`, `long`, `int`, `short`, `byte`, `char`, `int`.

Question 4. The rules are: (1) $\underline{\text{byte}} \rightarrow \underline{\text{short}} \rightarrow \underline{\text{int}} \rightarrow \underline{\text{long}} \rightarrow \underline{\text{float}} \rightarrow \underline{\text{double}}$ (2) $\underline{\text{char}} \rightarrow \underline{\text{int}} \rightarrow \underline{\text{long}} \rightarrow \underline{\text{float}} \rightarrow \underline{\text{double}}$.

Lesson page 6-4. Remarks about floating point

Question 1. The statement:

```
System.out.println( 100100100100100100.0 );
```

prints: `1.00100100100100096E17`.

Lesson page 6-5. Type char

Activity 6-5-1: Literals of type char

Question 1. True. A `char` literal begins and ends with a single quotemark.

Question 2. True. A `String` literal begins and ends with a double quotemark.

Question 3. An "escape sequence" is a literal that cannot be written as a single character. For example, '\t' denotes the tab character. All the Java escape sequences begin with the backslash (the "\") character.

Question 4. Here are the literals for various symbols:

Java literal	Symbol it represents
`'\\'`	backslash
`'\''`	single quote
`'\"'`	double quote
`'\n'`	line feed, or newline
`'\r'`	carriage return
`'\t'`	tab
`'\f'`	form feed
`'\b'`	backspace

Question 5. ASCII stands for "American Standard Code for Information Interchange". Each ASCII character uses one byte of memory.

Question 6. False. Java uses Unicode, not ASCII, to represent characters.

Activity 6-5-2: char as an integral type

Question 7. This works only for `0` $\leq$ `i` $\leq$ `9`. Look at the footnote on the lesson page to figure out why (or try it and see what happens).

```
char c= (char)(i + '0');
```

Question 8. Because type **int** is wider than **char**, no casting is required: **int** `i= c;`

Activity 6-5-3: A loop to sequence through characters

Question 9. In ASCII and Unicode, `'a'` is larger than `'A'`.

Question 10. **char** `capC= ('A'-'a') +'c';`

Activity 6-5-4: Execution of the loop

Question 11. The statement `c++;` increments character `c`.

Question 12. The expression is: `'A' <= c && c <= 'Z';`

Lesson page 6-6. Type `boolean`

Activity 6-6-1: The literals and operations of type `boolean`

Question 1. The literals **true** and **false**.

Question 2. Not, negation, and complement.

Question 3. Here's the truth table:

`b`	`c`	`b&&c`	`b||c`	`!b`	`b==c`	`b!=c`
true	**true**	**true**	**true**	**false**	**true**	**false**
true	**false**	**false**	**true**	**false**	**false**	**true**
false	**true**	**false**	**true**	**true**	**false**	**true**
false	**false**	**false**	**false**	**true**	**true**	**false**

Question 4. Here are the precedences:

1. Unary operators: `+ - ++ -- !`
2. Binary arithmetic operators: `* / %`
3. Binary arithmetic operators: `+ -`
4. Arithmatic relations: < > <= >=
5. Equality relations: `== !=`
6. Logical and: `&&`
7. Logical or: `||`

Question 5. We start with the formula and simplify it:

```
  (true || false) || 5 < 3 + 2
= true || 5 < 5
= true
```

Question 6. Equivalence is the boolean operator "==" (also called "equal to").

Question 7. Inequivalence is the boolean operator "!=" (also called "not equal to").

Activity 6-6-2: Short-circuit evaluation

Question 8. The manner in which conjunctions && and disjunctions || are evaluated. If the value of the operation can be determined from the first operand, the second is not evaluated. For example, in the boolean expression:

```
6 > 5 || 4 > 5
```

since the first operand, `6 > 5`, is true, only the first would be evaluated. However, with the following example, even though the first operand is true, both would have to be evaluated.

```
6 > 5 && 6 > 5
```

Question 9. When the first operand is **false**.

Question 10. When the first operand is **true**.

Question 11. This expression can be evaluated; since the first operand is **true**, the second operand is not evaluated, and the result of evaluation is **true**.

Question 12. This expression cannot be evaluated. Since `x = 0`, the first operand is **true**, the second operand is evaluated, and a divison by `0` occurs.

Activity 6-6-3: Properties of `boolean` operators

Question 13. The formula is: `(X && Y) == (Y && X)`

Question 14. The formula is: `(A || (B || C)) == ((A || B) || C)`

Question 15. The formula is: `(A && A) == A`

Question 16. `B || !B`, and `!(B && !B)`.

Question 17. `!(B && C) == (!B || !C)`, and `!(B || C) == (!B && !C)`.

Question 18. By DeMorgan's law (`!(B || C) == (!B && !C)`) and double negation (`!!B == B`), the formula is equivalent to:

```
(b && (i < j)) == (b && i > j)
```

This is true iff `b` is false, so it reduces to `!b`. So it is cannot be said to be true or false without knowing the value of `b`.

Activity 6-6-4: Exercises on type `boolean`

Activity 6-6-5: The mark of a boolean tyro

Question 19. A tyro is one (a person) familiar with the rudiments (basics) of a subject but lacking in practical experience.

Question 20. Three marks of a boolean tyro are:

1. **if** `(atHome ==` **true**`)` ...
2. **if** `(!atHome ==` **false**`)` ...

3.
   ```
   if (atHome || atWork)
         { b == true; }
   else { b == false; }
   ```

Question 21. Here we go:

```
     !((a != true) && (a == true)) == true
=    !(!a && a) == true
=    !(!a && a)
```

$$
\begin{aligned}
&= \texttt{!}\mathbf{false} \\
&= \mathbf{true}
\end{aligned}
$$