Chap02Yourname.java

Variables and operators

2.1 & 2.2 Declaration, assignment, and String

```
public class Chap02 Yourname{
        public static void main(String[] args) {

                System.out.println("2.1 & 2.2 Declaration, assignment, and String");
                System.out.println();
        }
}
```

2.1 & 2.2 Declaration, assignment, and String

- A **variable** is a named location that stores a **value**
- **Types** of variables:
  - String
  - float
  - double
  - byte
  - short
  - int
  - long
  - boolean
  - char

2.1 & 2.2 Declaration, assignment, and String

Name	*Range*	*Storage Size*	
`byte`	-2^7 to $2^7 - 1$ (-128 to 127)	8-bit signed	`byte` type
`short`	-2^{15} to $2^{15} - 1$ (-32768 to 32767)	16-bit signed	`short` type
`int`	-2^{31} to $2^{31} - 1$ (-2147483648 to 2147483647)	32-bit signed	`int` type
`long`	-2^{63} to $2^{63} - 1$	64-bit signed	`long` type
	(i.e., -9223372036854775808 to 9223372036854775807)		
`float`	Negative range: $-3.4028235E + 38$ to $-1.4E - 45$	32-bit IEEE 754	`float` type
	Positive range: $1.4E - 45$ to $3.4028235E + 38$		
`double`	Negative range: $-1.79769313486231 57E + 308$ to $-4.9E - 324$	64-bit IEEE 754	`double` type
	Positive range: $4.9E - 324$ to $1.79769313486231 57E + 308$		

2.1 & 2.2 Declaration, assignment, and String

```
String message;
//This statement DECLARES that the variable
//message has the type String.
message = “Hello, world!”;
//This is an assignment statement.
System.out.println(message);
```

2.1 & 2.2 Declaration, assignment, and String

```
String message = “Hello, world!”
System.out.println(message);
//DECLARATION and ASSIGNMENT in the same line.

message = “My name is Dubos.”;
System.out.println(message);
```

2.1 & 2.2 Declaration, assignment, and String

- Java keywords - https://docs.oracle.com/javase/tutorial/java/nutsandbolts/_keywords.html

abstract	continue	for	new	switch
assert***	default	goto*	package	synchronized
boolean	do	if	private	this
break	double	implements	protected	throw
byte	else	import	public	throws
case	enum****	instanceof	return	transient
catch	extends	int	short	try
char	final	interface	static	void
class	finally	long	strictfp**	volatile
const*	float	native	super	while

2.8 Order of operations

System.out.println(“2.8 Operators for String”);

System.out.println();

2.8 Order of operations

```
System.out.println(1 + 2 * 3);
System.out.println((1 + 2) * 3);
System.out.println(1 + 2 + “Hello”);
System.out.println(“Hello” + 1 + 2);
```

2.2 Initialization and int

```
System.out.println("2.2 Initialization and int");

System.out.println();
```

2.2 Initialization and int

- Variables must be **initialized** (assigned for the first time).

```
int hour;
System.out.println(hour);
```

2.2 Initialization and int

- Variables must be **initialized** (assigned for the first time).

```
int hour = 13;
System.out.println(hour);
```

2.2 Initialization and int

- Variables must be **initialized** (assigned for the first time).

```
int hour;
int minute;
int second;
hour = 13;
minute = 24;
second = 07;
```

2.2 Initialization and int

- Variables must be **initialized** (assigned for the first time).

```
int hour = 13, minute = 24, second = 07;
```

OR

```
int hour, minute, second;
hour = 13; minute = 24; second = 07;
```

2.2 Initialization and int

- Variables must be **initialized** (assigned for the first time).

- The current time is 13:24:7.

```
System.out.print(“The current time is “);
System.out.print(hour + “:” + minute + “:”);
System.out.println(second”.”);
```

2.2 Initialization and int

- Variables must be **initialized** (assigned for the first time).

- The current time is 13:24:7.

```
System.out.print(“The current time is “);
System.out.print(hour + “:” + minute + “:”);
System.out.println(second + ”.”);
```

2.2 Octal

```
System.out.println("2.2 Octal");

System.out.println();
```

2.2 Octal

- The current time is 13:24:8.

```
hour = 13, minute = 24, second = 08;
System.out.print(“The current time is “);
System.out.print(hour + “:” + minute + “:”);
System.out.println(second“.”);
```

2.2 Octal

- The current time is 13:24:8.

```
hour = 13; minute = 24; second = 08;
System.out.print(“The current time is “);
System.out.print(hour + “:” + minute + “:”);
System.out.println(second + “.”);
```

2.2 Octal

- The current time is 13:24:8.

- In Java and several other languages, an integer literal beginning with 0 is interpreted as an octal (base 8) quantity.

- Binary (base 2)
- Decimal (base 10)
- Hexadecimal (base 16)

2.2 Octal

- The current time is 13:24:8.

- In Java and several other languages, an integer literal beginning with 0 is interpreted as an octal (base 8) quantity.

- $(1)_2$
- $(10)_2$
- $(11)_2$
- $(1\ 0101\ 1111)_2$

2.2 Octal

- The current time is 13:24:8.

- In Java and several other languages, an integer literal beginning with 0 is interpreted as an octal (base 8) quantity.

- $(1)_2 = 1$
- $(10)_2 = 2$
- $(11)_2 = 3$
- $(1\ 0101\ 1111)_2 = ?$

2.2 Octal

- The current time is 13:24:8.

- In Java and several other languages, an integer literal beginning with 0 is interpreted as an octal (base 8) quantity.

- $(1)_2 = 1$
- $(10)_2 = 2$
- $(11)_2 = 3$
- $(1\ 0101\ 1111)_2 = 351$

2.2 Octal

- 8

System.out.println(0??????);

2.2 Octal

- ???

```
System.out.println(0123);
```

2.2 Octal

- The current time is 13:24:8.

```
hour = 13; minute = 24; second = 8;
System.out.print(“The current time is “);
System.out.print(hour + “:” + minute + “:”);
System.out.println(second + “.”);
```

2.5 Arithmetic operators, integer division

```
System.out.println("2.5 Arithmetic operators, integer division");

System.out.println();
```

2.5 Arithmetic operators, integer division

- Addition +
- Subtraction –
- Multiplication *
- Division /

2.5 Arithmetic operators, integer division

- Number of minutes since midnight: ???

```
System.out.print("Number of minutes since midnight: ");
System.out.println(                                    );
```

2.5 Arithmetic operators, integer division

- Number of minutes since midnight: ???

```
System.out.print("Number of minutes since midnight: ");
System.out.println(hour * 60 + minute);
```

- **Expression**
- The values operators work with are called **operands.**

2.5 Arithmetic operators, integer division

- Fraction of the hour that has passed: ???

2.5 Arithmetic operators, integer division

- Fraction of the hour that has passed: ???

```
System.out.print("Fraction of the hour that has passed: ");
System.out.println(minute / 60);
```

2.5 Arithmetic operators, integer division

- Fraction of the hour that has passed: ???

```
System.out.print("Fraction of the hour that has passed: ");
System.out.println(minute / 60);
```

- Java performs "**integer division**" when the operands are integers.
- By design, integer division always rounds towards zero.

2.5 Arithmetic operators, integer division

```
System.out.println(-4 / 3);
System.out.println(-3 / 3);
System.out.println(-2 / 3);
System.out.println(-1 / 3);
System.out.println(1 / 3);
System.out.println(2 / 3);
System.out.println(3 / 3);
System.out.println(4 / 3);
System.out.println(5 / 3);
System.out.println(6 / 3);
```

2.6 The floating-point type: double

System.out.println(“2.6 The floating-point type: double”);

System.out.println();

2.6 The floating-point type: double

Name	*Range*	*Storage Size*	
`byte`	-2^7 to $2^7 - 1$ (−128 to 127)	8-bit signed	`byte` type
`short`	-2^{15} to $2^{15} - 1$ (−32768 to 32767)	16-bit signed	`short` type
`int`	-2^{31} to $2^{31} - 1$ (−2147483648 to 2147483647)	32-bit signed	`int` type
`long`	-2^{63} to $2^{63} - 1$	64-bit signed	`long` type
	(i.e., −9223372036854775808 to 9223372036854775807)		
`float`	Negative range: −3.4028235E + 38 to −1.4E − 45	32-bit IEEE 754	`float` type
	Positive range: 1.4E − 45 to 3.4028235E + 38		
`double`	Negative range: −1.79769313486231 57E + 308 to −4.9E − 324	64-bit IEEE 754	`double` type
	Positive range: 4.9E − 324 to 1.79769313486231 57E + 308		

2.6 The floating-point type: double

- Java performs "**floating-point division**" when one or more operands are double values.

```
System.out.println(-4 / 3.0);
System.out.println(-3 / 3.0);
System.out.println(-2 / 3.0);
System.out.println(-1 / 3.0);
System.out.println(1 / 3.0);
System.out.println(2 / 3.0);
System.out.println(2 / 3.0);
```

2.6 The floating-point type: double

- Fraction of the hour that has passed: ???

2.6 The floating-point type: double

- Fraction of the hour that has passed: ???

```
System.out.print("Fraction of the hour that has passed: ");
System.out.println(minute / 60.0);
```

2.6 The floating-point type: double

```
double y = 1 / 3;
System.out.println(y);
```

2.6 The floating-point type: double

```
double y = 1 / 3.0;
System.out.println(y);
```

2.7 Rounding errors

System.out.println("2.7 Rounding errors");

System.out.println();

- Binary approximation of 0.1:
  - $(0.1)_{10} \approx (0.0001100110011001100110011001100)_2 = \frac{209715}{2097152}$
  - http://mathworld.wolfram.com/RoundoffError.html

2.7 Rounding errors

- Most floating-point numbers are only approximately correct.

```
System.out.println(0.1);
System.out.println(0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1);
System.out.println(0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1);
```

2.7 Rounding errors

2.7 Rounding errors

2.7 Rounding errors

2.7 Rounding errors

2.7 Rounding errors

- Over time the result may be inaccurate.
- We can avoid the problem by using integers in addition and subtraction, as long as the integer is smaller than

- 2,147,483,648 = 2^31

2.10 Types of errors

```
System.out.println("2.10 Types of errors");

System.out.println();
```

2.10 Types of errors

- **Compile-time error**

```
System.out.println(7)
```

2.10 Types of errors

- **Run-time errors**

```
System.out.println(7/0);
```

Exercise 2.2

```
System.out.println("Exercise 2.2");

System.out.println();
```

Exercise 2.3

```
System.out.println("Exercise 2.3");

System.out.println();
```