CSE 215: Programming Language II Lab
Sec – 8, Faculty - MUO
Lab Officer: Tanzina Tazreen
Lab – 1
Part: 2

Objective:
- Printing Text
- Data Types / Variables,
- Type Casting
- Printing Data,
- User Input, Operators,
- Conditional Statements,
- Switch

Printing Text:

```
System.out.print("Hello World!");

System.out.println("Hello World!");  //print from a newline in every
statement
```

Data Types / Variables:

JAVA Data types are divided into two groups:

- Primitive data types - byte, short, int, long, float, double, Boolean and char
- Non-primitive data types - String, Arrays etc.

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

Java Type Casting:

two types of casting:

- **Widening Casting** (automatically) - converting a smaller type to a larger type size byte -> short -> char -> int -> long -> float -> double

```
int myInt = 9;

double myDouble = myInt;
```

- **Narrowing Casting** (manually) - converting a larger type to a smaller size type
  double -> float -> long -> int -> char -> short -> byte

```
double myDouble = 9.78;

int myInt = (int) myDouble;
```

Printing out data

Java uses the + operator to perform string concatenation. That means, you can print out values of variables within a System.out.println() call.

```
System.out.println("myInteger contains = " + myInteger);
```

Here, Java sees that the first operand of the + operator is a string. Then, it converts the rest of the operands to string and concatenates them together to form one string. That string is printed out to the console. The above line of code should print out the following line:

```
>>> myInteger contains = 5
```

The same is true for most other types of variables (at least the primitive datatypes).

```
System.out.println("myFloat contains = " + myFloat);
System.out.println("myChar contains = " + myChar);
```

You can chain together multiple variables and strings as well.

```
System.out.println("myDouble contains = " + myDouble + " and myString contains = " + myString)
```

Taking User Input

```
1  import java.util.Scanner;
2
3  public class InputTest {
4    public static void main(String[] args) {
5      Scanner input = new Scanner(System.in);
6
7      System.out.println("Please enter an integer");
8      int userGivenInteger = input.nextInt();
9
10     System.out.println("You entered: " + userGivenInteger);
11   }
12 }
```

Program 1: Taking an integer as user input and printing the obtained value out to the console

The above program has a lot going on. Let's break it down:

Line 1: We import the built-in `Scanner` class from `java.util` package. There are a lot of modules built into Java, similar to header files in the C programming language. Think of this as analogous to importing a header file in C programming language.

Line 5: This will become clearer when we cover Object Oriented Programming concepts.

However, the main thing to know here is that we are instantiating an object which is of type `Scanner`. This will enable us to access the methods available to the `Scanner` class.

Line 7: We print out a prompt for the user so that s/he can type an input integer.

Line 8: We use one of the methods of the Scanner class, `nextInt()` , to obtain the next

integer the user enters to the console.

To take various variables as input, you need to use the appropriate methods for it. Table 1 summarizes the appropriate methods.

Datatype	Method Name
`Integer/int`	`nextInt()`
`Double/double`	`nextDouble()`
`Float/float`	`nextFloat()`
`String`	`nextLine() for strings with spaces, next() for words`
`Boolean/boolean`	`nextBoolean()`

Table 1: Data Types and their corresponding method in Scanner class for user input.

Boolean Expressions:

Boolean Expressions are formed using boolean operators and boolean values comparison operators.

Comparison Operators		Boolean Operators	
Operator	**Name**	**Operator**	**Name**
==	Equal to	!	NOT
!=	Not equal to	&&	AND (both)
<	Less than	||	OR (any one or both)
>	Greater than	^	Exclusive OR (any one but **not** both)
<=	Less than or equal to		
>=	Greater than or equal to		

Table 1: Types of Comparison and Boolean operators

Conditional statements

Based on some condition, execute a select block of code.

if-else	**switch case**
Standard conditional statement	Useful if your code turns out to have multiple "else if" clauses with equal to comparison

`if (condition) {`
`  // code`
`} else {`
`  // code`
`}  if`
`(condition1) {`
`  // code`
`} else if (condition2) {  // code`
`} else if (condition3) {`
`  // code`
`}  ... else if (conditionN) {`
`  // code  }`
`else {`
`  // code`
`}`
	`switch (conditionVar) {  case 1:  //`
	`code  break;  case 2:  //`
	`code  break;  case n:`
	`// code`
	`break;`
	`  default:  // code`
	`break;`
	`}`

Table 2: Types of if-else statements.

Task:

1. Write a program that takes an integer and determines if it's odd or even. Use switch cases to produce result.

2. Write a program that takes an integer and determines if it's prime or not. A number is prime if it is divisible by 1 and itself only, i.e. 2, 3, 11, 37 etc.

3. Consider the given BMI ranges

If your BMI is:
below 18.5 – you're "underweight" between
18.5 and 24.9 – you're "healthy" between 25
and 29.9 – you're "overweight" between 30
and 39.9 – you're "obese"

Write a program that takes a decimal value as input from the user. Then print the quoted words above based on the range.

(a) Use if-else if-else

(b) Use ternary operator

4. Take a year as user input. Then print it check if it's a leap year or not.

Note: A leap year must satisfy **any or both** of the following conditions:

Divisible by 400

Divisible by 4 and not divisible by 100

Sample output:

2015: false