UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Software Development II

Lecture 2 – Data Types and Control Structures

*Reading : **Java for Everyone** - Chapter 02 / Chapter 03*

I HEAR, I KNOW. I SEE, I REMEMBER. I DO, I UNDERSTAND.

CONFUCIUS

From Last Week

- How Java code is executed?
- Java Class file : Byte Code
- A Simple Java Application

```
// HelloWorld.java Our first Java Application

class HelloWorld
{
      public static void main( String args[])
      {
            System.out.println( "Hello World!" );
      }
}
```

- Purpose of main method
- Concept of packages
- Input Reading using packages

Today’s Outline

Data types

- Variables
  - Declaration
  - Strings
  - Booleans
  - Operators
  - Casting
- Constants

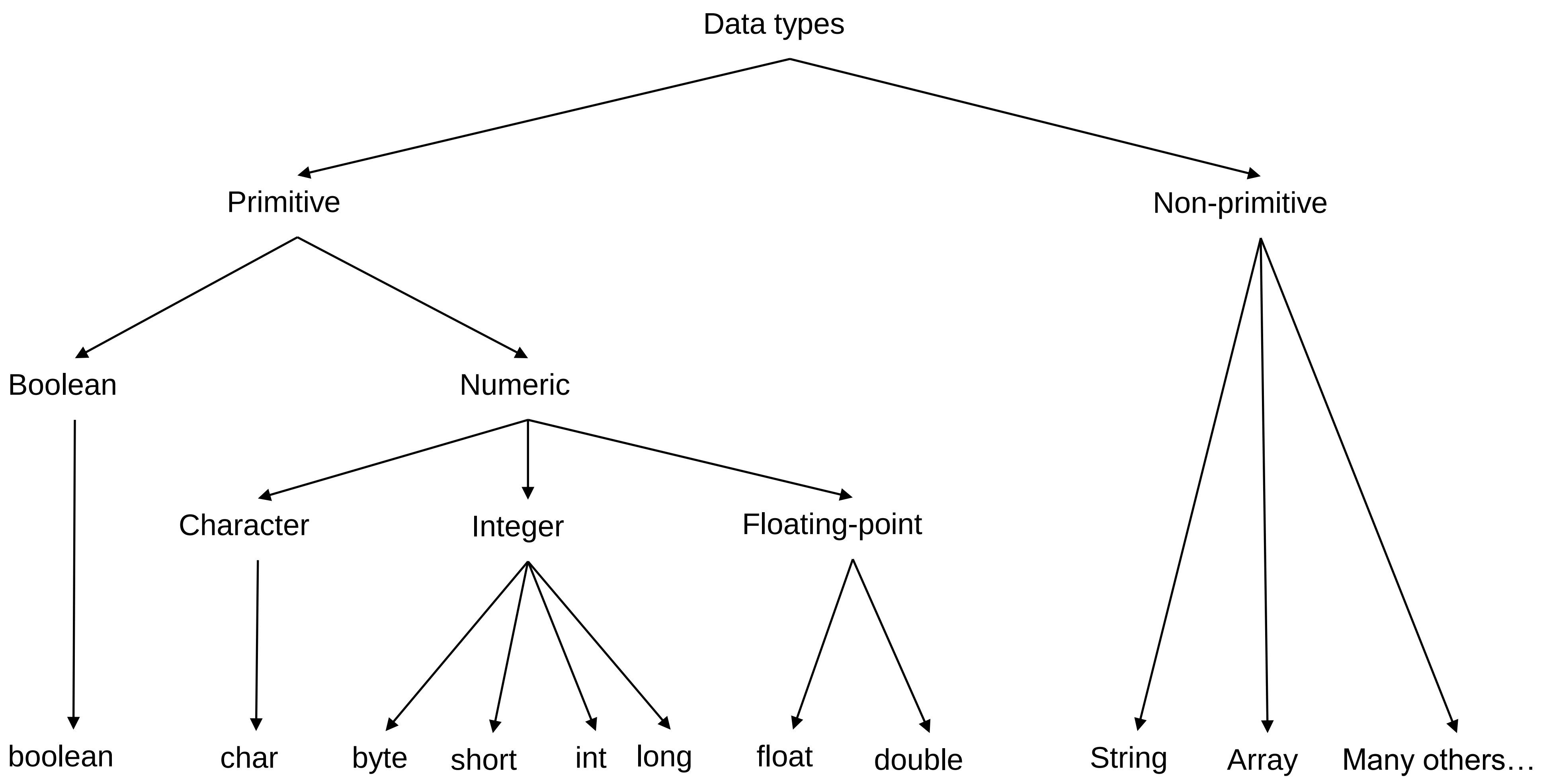
Control structures

- Conditionals
  - If
  - If-else
  - Nested if
  - If-else ladder
  - Switch Case

Data Types

In Java programming, data types define the type of data that a variable can hold. **Java is a statically-typed language**, which means that the type of a variable must be declared before it is used.

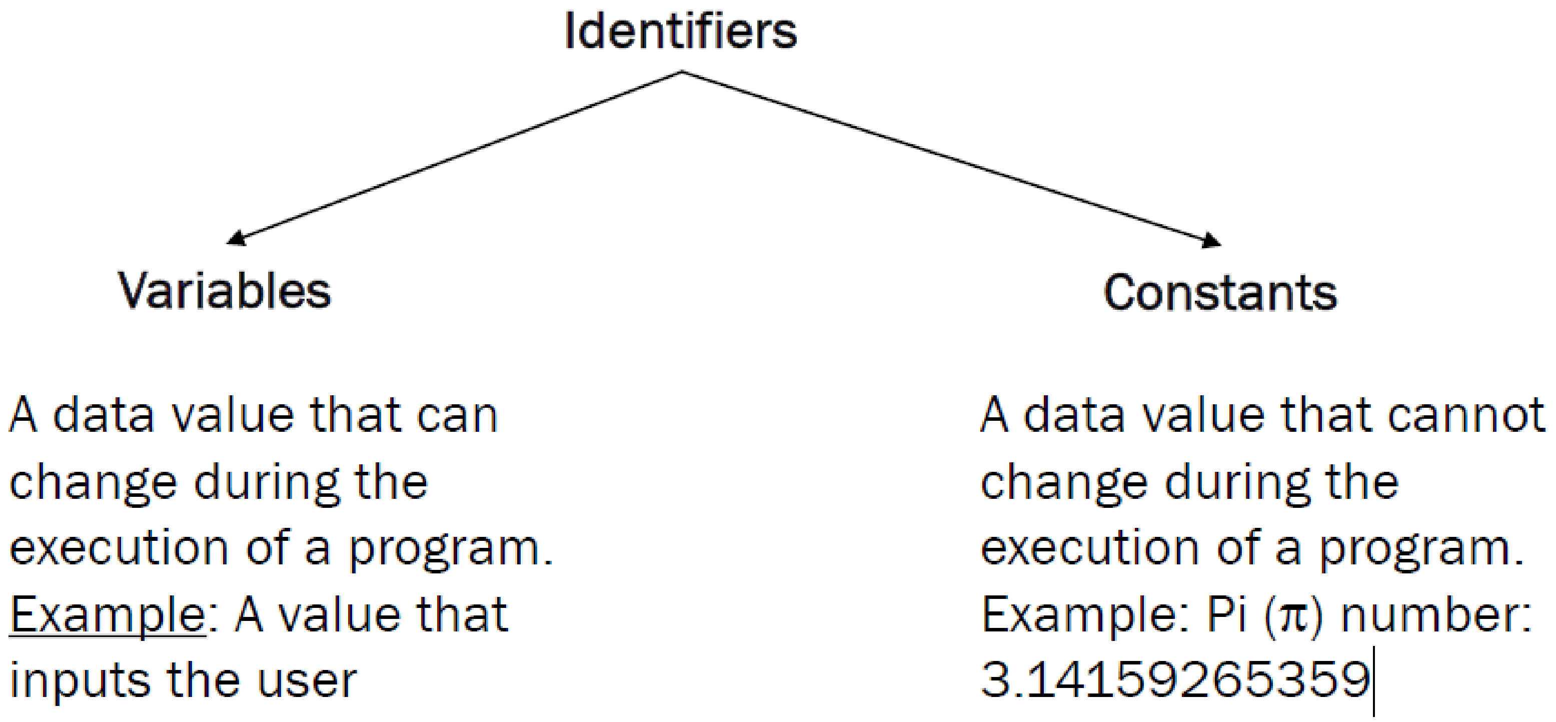
Data Types



Data types
Primitive
Non-primitive
Boolean
Numeric
Character
Integer
Floating-point
boolean
char
byte
short
int
long
float
double
String
Array
Many others…

Data types

There are two types of identifiers



A data value that can change during the execution of a program.
Example: A value that inputs the user

A data value that cannot change during the execution of a program.
Example: Pi (π) number: 3.14159265359

Data Types

- In Java we **always** need to specify the data type.

Type	Description	Size	Range	Example
boolean	True or False	1 bit	true, false	`boolean bool = true;`
byte	Integer	8 bits	-128 to 127	`byte b = 10;`
char	Character	16 bits	ASCII values from 0 to 255	`char letter_a = ‘a’;`
short	Integer	16 bits	-32,768 to 32,767	`short val = 30000;`
int	Integer	32 bits	-2,147,483,648 to 2,147,483,647	`int val = 10000;`
long	Integer	64 bits	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807	`long val = -9800L;`
float	Floating point	32 bits	Up to 7 decimal digits	`float f = 1.23f;` `float f = 1.23e10f;`
double	Floating point	64 bits	Up to 16 decimal digits	`double d1 = 1.23d;` `double d2 = 1.233e300d;`

Variable declaration (creation) and initialization: (syntax)

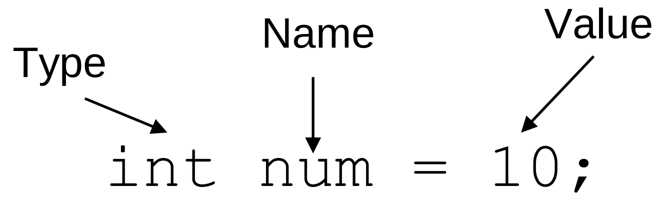
- When declaring a variable in Java, you need to specify:
  - The type
  - The name

    declaration
  - The value (optional) => initialization

```
<data type> <variable name>;

<data type> <variable name> = <value>;
```

  - Declaration
    - **`int number;`**
  - Initialization
    - **`number= 25;`**
  - Declaration and initialization
    - **`int number=25;`**



```
int num = 10;
```

Variable declaration (creation): example

- Examples
  - `int number = 30;`
  - `char letter = ‘a’;`
  - `boolean ready = true;`

- Always use informative variables names:
  - `int a = 10;char b = ‘a’;` //No Syntax errors but not informative
  - `int number = 10;`
  - `boolean response= true;` //Informative and meaningful

Booleans

- A Boolean is a non-numerical primitive data type.
- A Boolean can only have two values: **true** or **false**(logical values)
- Declaration and initialization:

  - `boolean is_true= true;`
  - `boolean is_false= false;`

- A Boolean variable declared with no initialistion will have a default value of false.

- `Boolean my_boolean; // this variable is false`

Strings

Strings are a non-primitive data type used to store text.

Sequence of characters separated by double quotes.

- Declaration: **`String <name> = “value”;`**

  `String name = “Alan”;`
- find the length of a string : **`int length = name.length();`**
- Convert to upper case: **`String upper_case = name.toUpperCase();`**
- Convert to lower case: **`String lower_case = name.toLowerCase();`**
- Concatenation: **`String twice = name.concat(name);`**
  - Read more at : https://www.programiz.com/java-programming/string
- Numbers and strings: be careful when concatenating numbers and strings.
- Special characters: You can print special characters using the escape character “\” before them: **`System.out.println("These are special characters: \" \\");`**

Input with data type

- When you want to read a specific data value from the user, you can use the type that you need.
- Example with int:

```
int integer = input.nextInt()
```

- Example with String:

```
String word = input.next();
```

- Example with double:

```
double height = input.nextDouble();
```

Type Casting

- Casting happens when a variable of one type gets converted to another type.
- When a smaller data type is assigned to a larger data type, the conversion is automatic:
  - **`int int_number = 10;`**
  - **`double double_number = int_number;`**

- When a larger data type needs to be converted into a smaller data type, you will need to do casting:
  - **`double num_1 = 2.5d;`**
  - **`int num_2 = (int) num_1;`** → we place the new type in paranthesis in from the variable

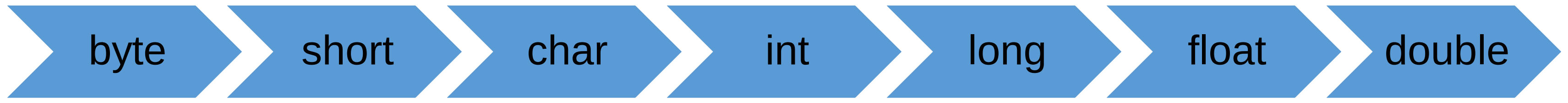
This is an int

Casting to int

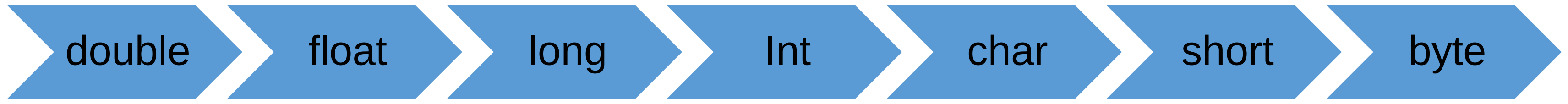
This is double

Casting

- Auto casting (implicit) when the new type is larger:



- Manual casting (explicit) when the new type is smaller



- Automatic casting can happen without you realizing:

-
  ```
  int value = 10/3;
  System.out.println(value);
  ```
  This will print 3

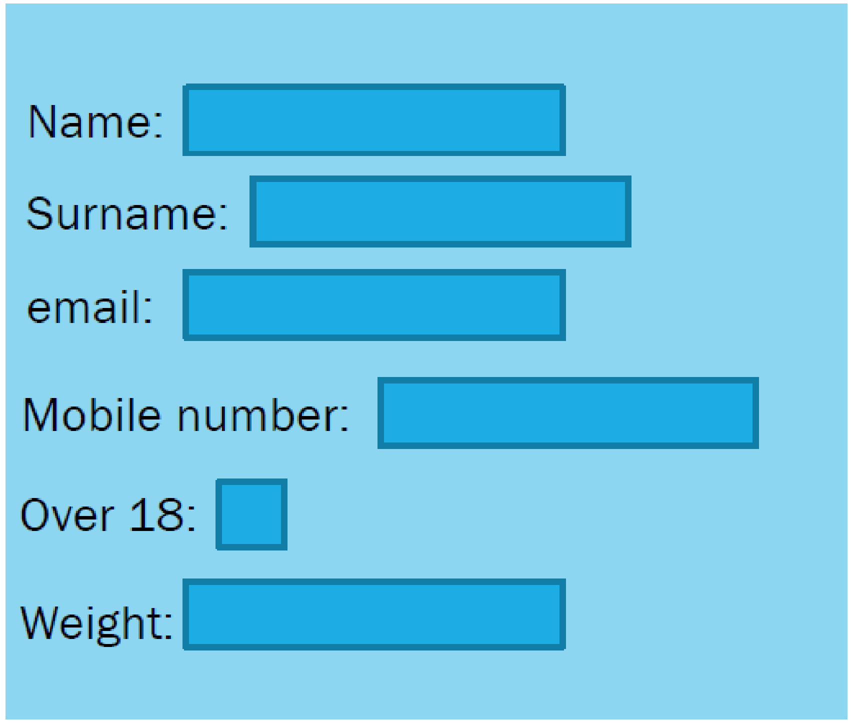
-
  ```
  double value = 10/3;
  System.out.println(value);
  ```
  This code will print 3.0.

-
  ```
  double value = 10/3.0;
  System.out.println(value);
  ```
  This code will print 3.3333333333333335

Selection of required data types

- Which data types would you use to implement the following form variables?

FORM



Name:	
Surname:	
email:	
Mobile number:	
Over 18:	
Weight:	

Operators : Arithmetic Operators

Operator	Name	Description	Example
+	Summation	Adds together two values	x + y
-	Subtraction	Subtracts one value from another	x - y
*	Multiplication	Multiplies two values	x * y
/	Division	Divides one value by another	x / y
%	Modulus	Returns the division remainder	x % y
++	Increment	Increases the value of a variable by 1	++x
--	Decrement	Decreases the value of a variable by 1	--x

Examples:

```
int num_1 = 10;
int num_2 = 20;
int num_3 = num_1 + num_2;
System.out.println(num_3);

String name = "Alan";
String surname = "Turing";
System.out.println("Hi " + name + " " + surname + ".");
```

Arithmetic Operator Precedence

- High priority *** / %**
- Low priority **+ -**
- Parenthesis contents are evaluated first!!
  - Left-to-right passes
  - Innermost to outer
- Expressions are evaluated from; left → right

  Example:

  74 / 10 % 2 * 5 – 10 % (5 – 1)

  - First deal with ( )
  - Next work from left to right on / , %and operators
  - Finally perform the subtraction

Operators : Assignment Operators

Operator	Description	Example	Same As
=	Assignment	x = 5	x = 5
+=	Assignment with summation	x += 3	x = x + 3
-=	Assignment with subtraction	x -= 3	x = x - 3
*=	Assignment with multiplication	x *= 3	x = x * 3
/=	Assignment with division	x /= 3	x = x / 3
%=	Assignment with modulus	x %= 3	x = x % 3

- Used to assign the value of an expression to a variable.
- Usual assignment operator **=**

Operators : Prefix and Postfix Operators

- Prefix operator: **y = ++m;** or **y = --m;**
  - Adds/subtracts **1** to the operand **m**
  - Result is assigned to the variable **y** on left

- Postfix operator: **y = m++;** or **y = m--;**
  - Assigns the value to the variable **y** on left
  - Increments/decrements the operand **m**

Exercises

What will be the final values of following variables. Expressions are executed individually.

```
int i = 3, j = 4, k = 5, l=0, m=0 ;
```

- `m = ++i ;`
- `l = j -- ;`
- `m = ++ k % -- j ;`
- `l = j ++ * -- i ;`
- `m = ++ j + i ;`

What will be the output of following code

```
int num_1 = 10;
Int num_2 = 20;
int num_ 3 = num_1 + num_2;
num_3 = num_3 * 2;
num_3 = num_3 + 2;
System.out.println(num_3);
```

Operators: Relational(Comparison) Operators

Operator	Name	Example
==	Equal to	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

$\longrightarrow$ **Not to be confused with '='**

- Used to compare two quantities, and depending on their relation to take decisions.
- Expressions containing relational operators are relational expressions.

Assignment vs Comparison operators

- `int num = 5;` $\longrightarrow$ Creates a variable called num and assigns the value 5
- `System.out.println(num = 6);` $\longrightarrow$ Assigns the value 6 to num and prints the variable with value 6
- `System.out.println(num == 5);` $\longrightarrow$ Compares variable `num` whith value 6 to 5. 6 and 5 are different therefore prints `false`. Note that after the second line of code, the value of `num` is 6.

- Output:
- `6`
- `False`

Operators: Logical Operators

Operator	Name	Description	Example
&&	Logical AND	Returns true if both statements are true	x < 5 && x < 10
||	Logical OR	Returns true if one of the statements is true	x < 5 || x < 4
!	Logical NOT	Reverse the result, returns false if the result is true	!(x < 5 && x < 10)

- Used to combine two or more relational expressions and such are called as logical expressions.
- Value of a logical expression - true or false
- Example

```
int num = 8;

System.out.println(num > 0 && num < 5);
```

`num > 0` is `true` and `num < 5` is `false`
`true && false` is `false`

Output:
```
False
```

Booleans with operators

- Booleans are often used with local operators
- `Boolean is_sunny= true;`
- `Boolean is_warm= false;`
- `Boolean sunny_and_warm= is_sunny`

- Booleans are also often used with assignment operators:
- `int x = 5;`

- `Boolean equal = (x == 5);` $\longrightarrow$ `true`
- `Boolean not_equal= (x == 0);` $\longrightarrow$ `false`
- `Boolean greater_than_0 = x > 0;` $\longrightarrow$ `true`
- `Boolean greater_than_20 = 10 > 20;` $\longrightarrow$ `false`

Guess the output

```
int num_1 = 5;
int num_2 = 20;
boolean bol_1 = num_1 > 10;
boolean bol_2 = num_2 > 10;
System.out.println(bol_1 || bol_2);
```

Logical Operators cont…

(condition1 **&&** condition2)

is true if and only if both condition1 and condition2 are true

(condition1 **||** condition2)

is true if and only if condition1 or condition2 (or both) are true

! condition1

is true if and only if condition1 is false

Constants

- A constant is a data type that cannot change (also considered as a type of variable).
- The use of constants can make your program safer (as the value will not be modified).
- To create a constant in Java, we use the modifier final:

final <data type> <name> = <value>;

Example:

```
      final float pi = 3.14f;
      pi=8.56 ❌; //you cannot change a constant
value
```

Control Structures

Control Flow Statements

- Control flow statements, break up the flow of execution by:
  - Decision making
  - Looping
  - Branching
- These are used to conditionally execute particular blocks of code.

Types of Control Flow Statements

- Decision-making/ Conditional statements:
  if, if-else, switch
- Looping statements:
  while, do-while, for
- Branching statements:
  break, continue, return

Conditionals: `if` Statements

- Use the `if` statement depending on the complexity of conditions to be tested.
  - Simple if statement
  - If-else statement
  - Nested if - else statement
  - else if ladder

Conditionals: *if*

If 🌧

I will take my ☂

```
if (condition) {
  // code to be executed if the condition is true
}
```

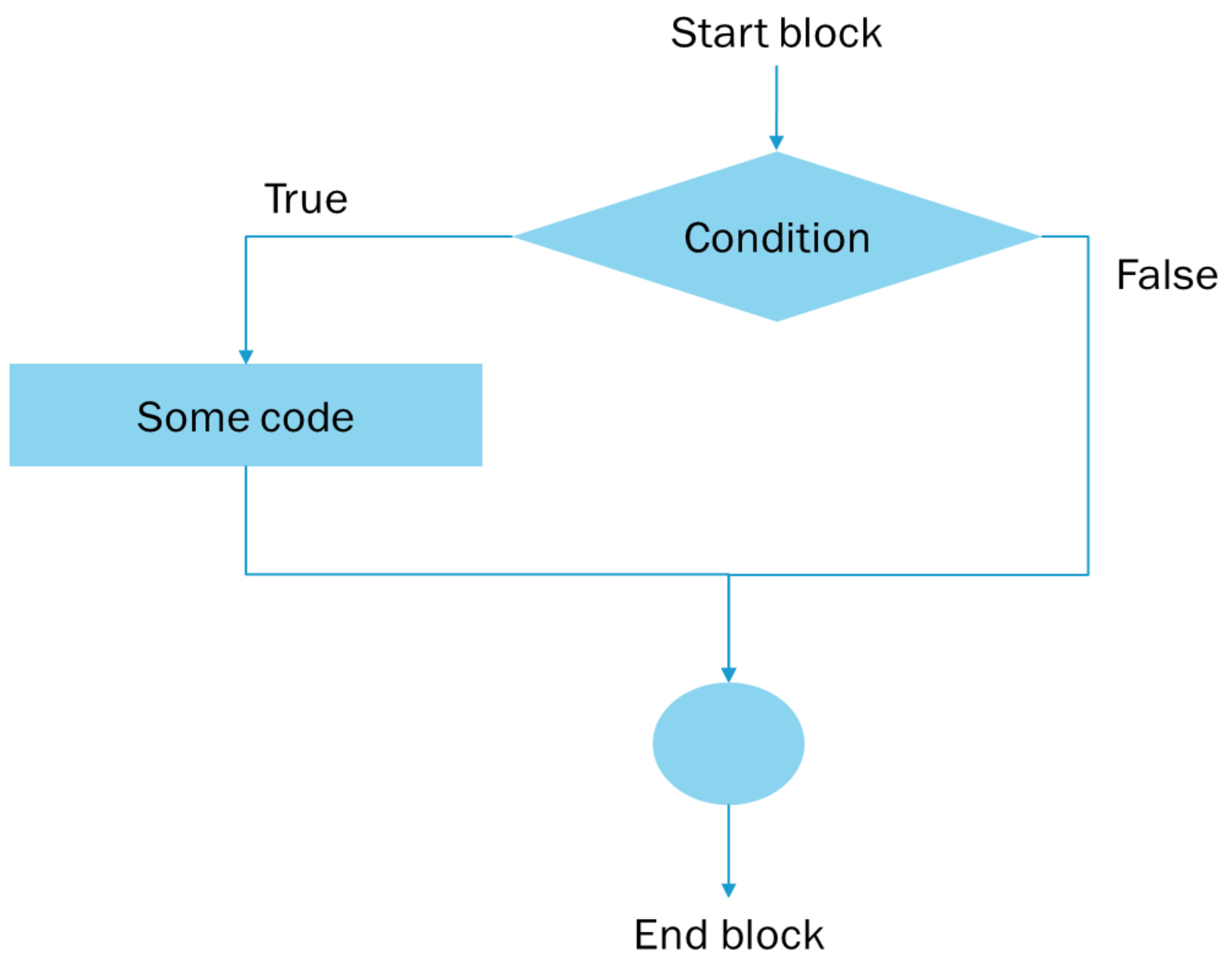
Today is **raining**. What will I do?

Tomorrow is **not raining**. What will I do?

```
boolean is_raining = true;

if (is_raining) {
     System.out.println("Take umbrella.");
}
```

Conditionals: *if*



Conditionals: *if .. else ..*

If 🌧

I will take my ☂

Otherwise

I will take my 🚴

```
if (condition) {
  // code to execute if the condition is true
}
else {
  // code to execute if the condition is false
}
```

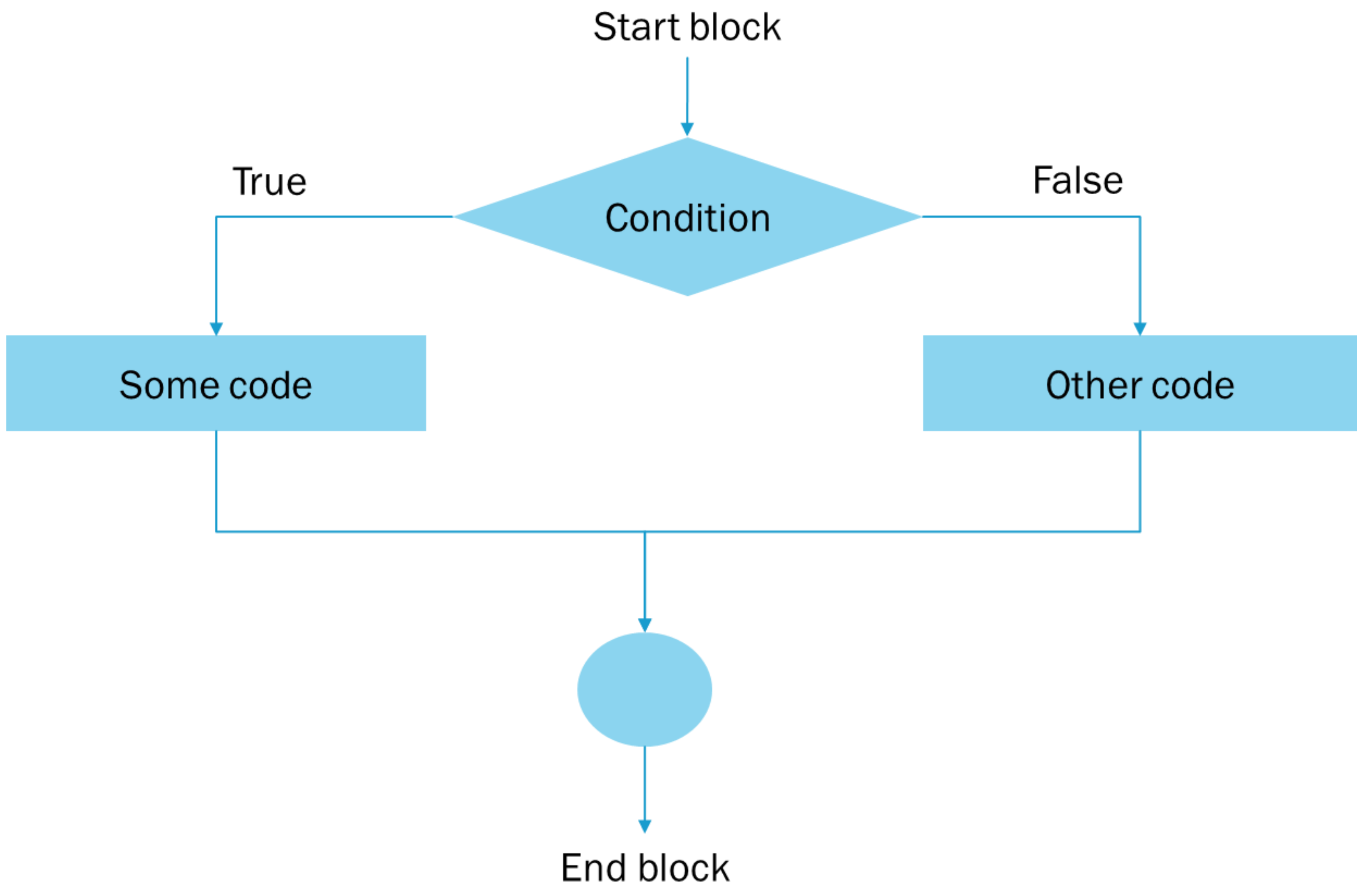
```
boolean is_raining = false;

if (is_raining) {
      System.out.println(“Take umbrella.”);
}
else{
       System.out.println(“Take bike.”);
}
```

Today is **not raining**. What will I do?

Today is **cloudy**. What will I do?

Conditionals: *if .. else ..*



Conditionals: *if .. else if .. else ..*

If

I will take my

Otherwise If

I will take the

Otherwise

I will take my

```
if (condition) {
  // code to execute if the condition is true
}
else if (condition){
  // code to execute if the 1st condition is false
  // and the second condition is true
}
else{
       // code to execute if the first and second
       // conditions are false
}
```

Today is **cloudy**. What will I do?

Today is **raining and cloudy**. What will I do?

Conditionals: *if .. else if .. else ..*

If

I will take my

Otherwise If

I will take the

Otherwise

I will take my

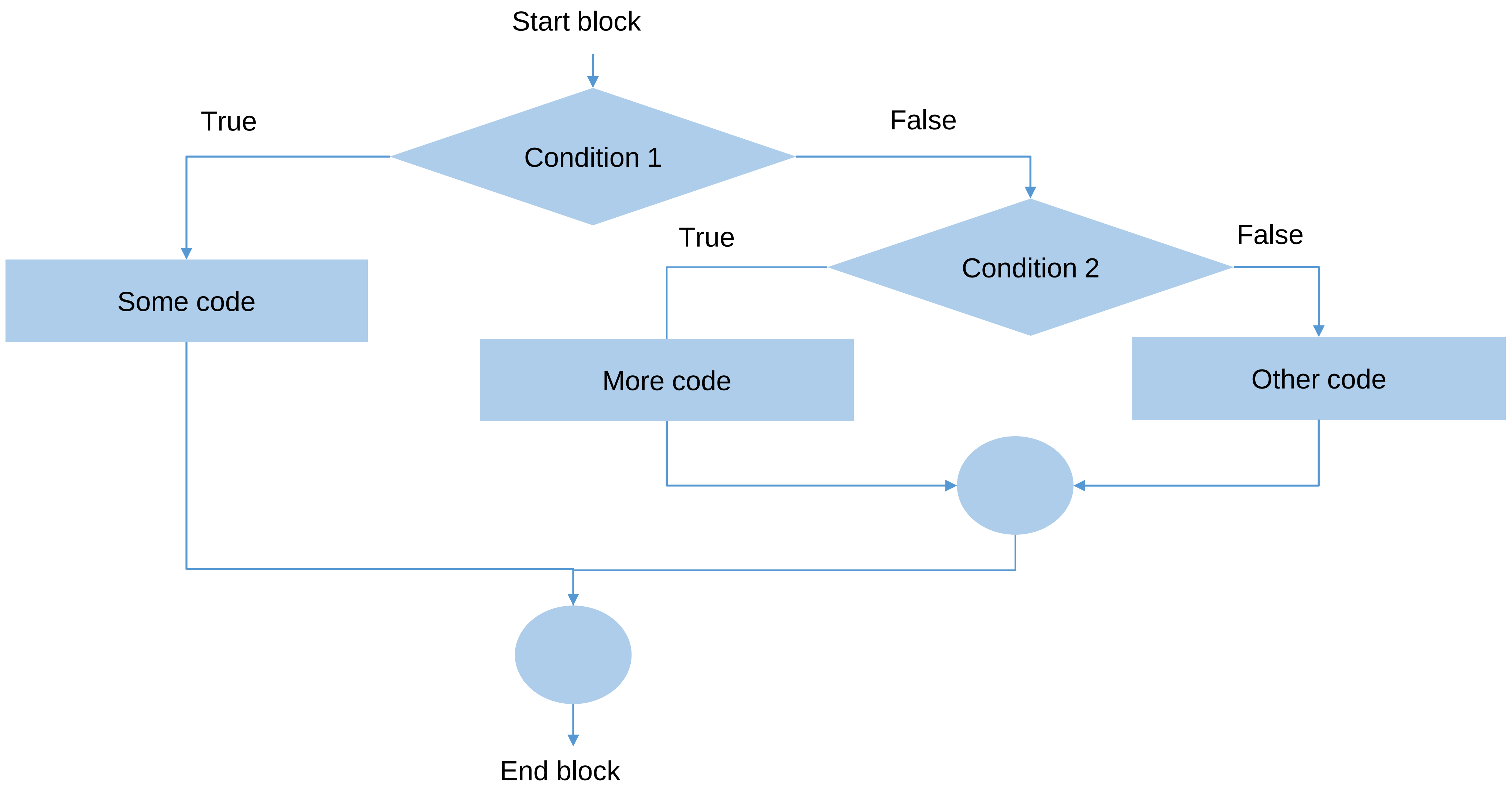
Today is **cloudy**. What will I do?

Today is **raining and cloudy**. What will I do?

```
boolean is_raining = false;
boolean is_cloudy = true;

if (is_raining) {
      System.out.println(‘Take umbrella.’);
}
else if (is_cloudy) {
       System.out.println(‘Take train.’);
}
else{
       System.out.println(‘Take bike.’);
}
```

Conditionals: *if .. else if .. else ..*



Difference between *if + if* and *if + else if*

- Which one is more efficient?

```
if (condition) {
  // code
}
else if (condition){
  // code
}
```

```
if (condition) {
  // code
}
if (condition){
  // code
}
```

- In a *if + else* statement, the condition of the second statement is only checked if the first condition is false.
- In a *if + if* statement, all conditions are checked.

Another example

If >= 40

I will go on

Otherwise

I will have to

I got a mark of 35. What will I do?

I got a mark of 78. What will I do?

```
int mark = 35;

if (mark >= 40) {
     System.out.println(“Holidays!”);
}
else{
     System.out.println(“I will
study.”);
}
```

Exercise - 01

- Print the grade of the student depending on the mark as per below requirements
- Grade “A”- if mark is more than 74
- Grade “B” – if mark is between 60-74
- Grade “C” – if mark is between 40-60
- Grade “F” – if mark is below 40
- Write the answer using simple if and if-else ladder

Nesting of `if-else` Statements

```
if ( <condition> ){
    if ( <sub condition> )
     < code block 1 >
    else
     < code block 2 >
}
else{
    if ( <sub condition> )
     < code block 3 >
    else
     < code block 4 >
}
```

Exercise: real case scenario

- We are managing a site that has a capacity of 100. We have sold (so far) 50 tickets and we want to sell another ticket. If there is space left, we will sell a ticket, otherwise we will say that is full.
- What is the correct condition here? Build the conditional logic.

```
int capacity = 100;
int seats_occupied = 50;

if (seats_occupied >= capacity  ) {
      System.out.println("Full.");
}
else
{
      seats_occupied++;
}
```

Exercise: real case scenario

- We have created a menu for a program that asks if the user wants to add a user (‘a’) or delete a user (‘d’).
- **What are the missing conditions here? Build the conditional logic.**

```
System.out.println("Enter ‘a’ to add and ‘d’ to delete.");
Scanner getInput = new Scanner(System.in);
String option = getInput.next();

if (option.equals(“a")) {
System.out.println("Adding user.");
}
else if (option.equals("c"))
{
System.out.println("Deleting user.");
}
else{
System.out.println("Invalid option.");
}
```

Comparing Strings

- Don't use `==` for strings!

```
    if (input == "Y") // WRONG!!!
```

- Use `equals()` method:

```
    if (input.equals("Y"))
```

- For case insensitive test ("Y" or "y") use `equalsIgnoreCase()`

```
    if (input.equalsIgnoreCase("Y"))
```

- `“s”.compareTo(“t”) < 0` means:
  `s` comes before `t` in the dictionary
- Read More at
  - https://dzone.com/articles/how-do-i-compare-strings-in-java

Statements with multiple conditions

- Check if the number that the user enters is between 0 and 10 (validation):

```
System.out.println("Enter a number between 0 and 10");
Scanner getInput = new Scanner(System.in);
int number = getInput.nextInt();

if (number >= 0)
{
System.out.println("Number larger than 0.");
}
if (number <= 10)
{
System.out.println("Number smaller than 10.");
}

if (number >= 0 && number <= 10)
{
System.out.println("Number larger than 0 and smaller than 10.");
}
```

Equivalent to

AND: Both conditions need to be true to print correct.

Statements with multiple conditions

- Example of a decision support system that makes a decision considering the weather and how we feel:

```
boolean im_tired = false;
boolean is_raining = true;

if (im_tired){
      System.out.println(“Take train.");
}

else if (is_raining){
      System.out.println(“Take train.");
}


if (im_tired || is_raining){
      System.out.println(“Take train.");
}
```

Equivalent to

OR: Only one needs to be true to print ‘Take train’.

The **switch** Statement

```
switch (variable)
{
  case value1:
         ...
          break;

  case value2:
         ...
          break;
     ...
     ...
    default:
           ...
          break;
}
```

switch

case

default

break

Reserved words

Don’t forget breaks!

The **switch** Statement

Instead of many *if* and *else* we can use a switch:

```
switch (expression) {
      case x:
            // code here
            break;
      case y:
            // code here
            break;
      default:
}x
```

```
String weather = "cloudy";
switch (weather) {
    case "rain":
        System.out.println("Take umbrella.");
        break;
    case "cloudy":
        System.out.println("Take train.");
        break;
    default:
        System.out.println("Take bike.");
}
```

Switch

- Switch is a great structure to use in a menu:

```
Scanner input = new Scanner(System.in);
System.out.println("* MENU *");
System.out.println("1.- Print user name");
System.out.println("2.- Add user");
System.out.println("3.- Delete user");
System.out.print("Choose an option: ");
int option = input.nextInt();

switch(option){
      case 1:
            // some code to print user name
            break;
      case 2:
      // some code to add a user
            break;
      case 3:
            // some code to delete a user
            break;
      default:
            System.out.println("Option selected is not correct.");
  }
```

(defined in the Jump section)

Rules that applies to a `switch` statement

- The variable used in a switch statement can only be a byte, short, int, or char.
- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon.
- The value for a case must be the same data type as the variable in the switch, and it must be a constant or a literal (character).
- When the variable being switched on is equal to a case, the statements following that case will execute until a *break* statement is reached.

Rules that applies to a `switch` statement

- When a *break* statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.
- Not every case needs to contain a break. If no break appears, the flow of control will *fall through* to subsequent cases until a break is reached.
- A *switch* statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

Switch:Missing break

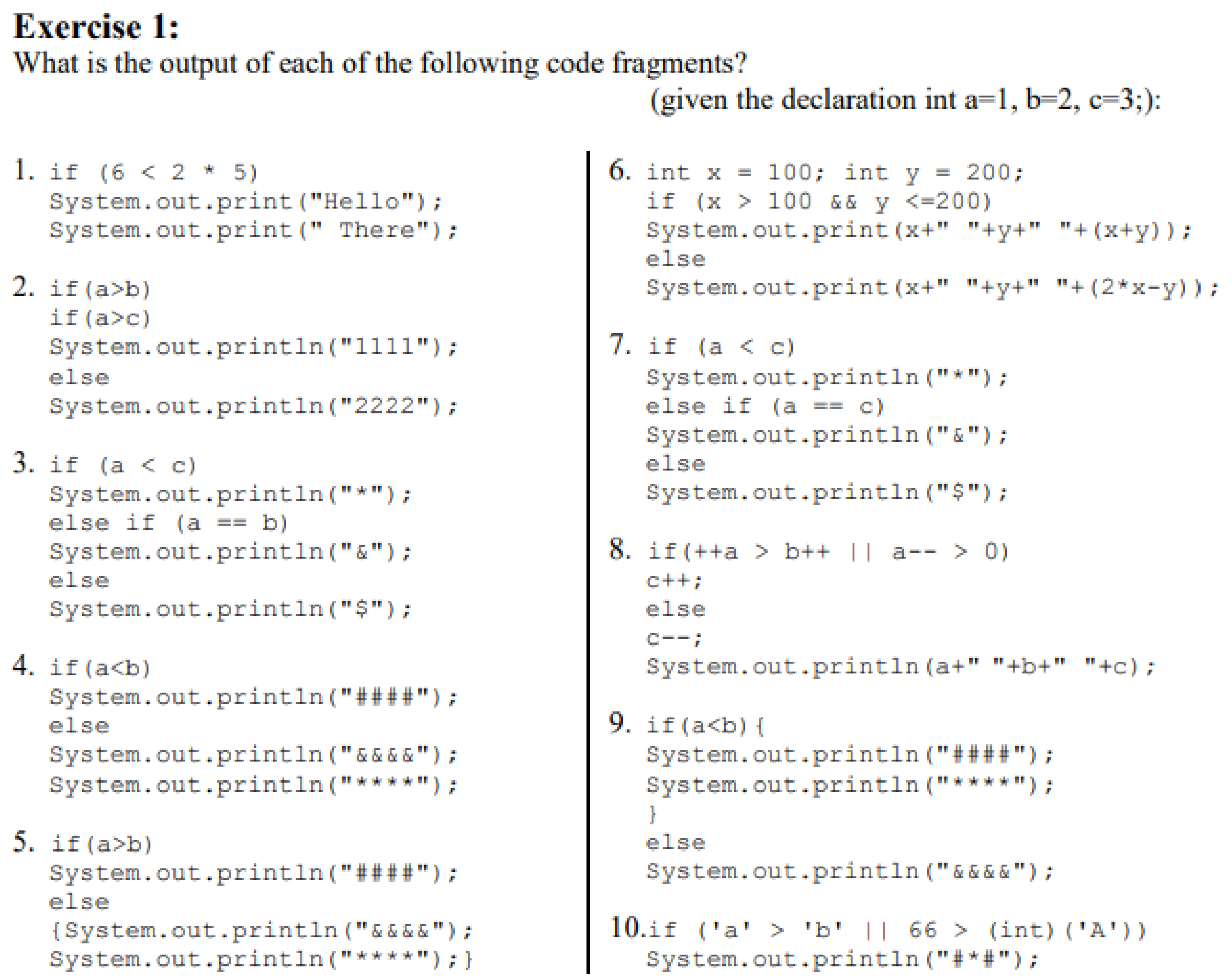
- Guess the result?

```
switch (num)
{
 case 1:
 case 2:System.out.println ("Buckle your shoe");
        break;
 case 3:
 ...
}
```

Exercise

- What is the value of `s.length()` if `s` is
  a. *the empty string* `""`*?*
  b. *the string* `" "` *containing a space?*
  c. ***`null`?***

Exercise : Homework



Exercise 1:
What is the output of each of the following code fragments?
(given the declaration int a=1, b=2, c=3;):

1.
```
if (6 < 2 * 5)
System.out.print("Hello");
System.out.print(" There");
```

2.
```
if(a>b)
if(a>c)
System.out.println("1111");
else
System.out.println("2222");
```

3.
```
if (a < c)
System.out.println("*");
else if (a == b)
System.out.println("&");
else
System.out.println("$");
```

4.
```
if(a<b)
System.out.println("####");
else
System.out.println("&&&&");
System.out.println("****");
```

5.
```
if(a>b)
System.out.println("####");
else
{System.out.println("&&&&");
System.out.println("****");}
```

6.
```
int x = 100; int y = 200;
if (x > 100 && y <=200)
System.out.print(x+" "+y+" "+(x+y));
else
System.out.print(x+" "+y+" "+(2*x-y));
```

7.
```
if (a < c)
System.out.println("*");
else if (a == c)
System.out.println("&");
else
System.out.println("$");
```

8.
```
if(++a > b++ || a-- > 0)
c++;
else
c--;
System.out.println(a+" "+b+" "+c);
```

9.
```
if(a<b){
System.out.println("####");
System.out.println("****");
}
else
System.out.println("&&&&");
```

10.
```
if ('a' > 'b' || 66 > (int)('A'))
System.out.println("#*#");
```

Exercise

- Convert the following switch statement into if-else statements

```
String dayString1, dayString2, dayString3;
int day = KB.nextInt();
switch (day) {
  case 1: dayString1 = "Saturday";
  case 2: dayString2 = "Sunday";
          break;
  case 3: dayString3 = "Monday";
          break;
  case 4: dayString1 = "Tuesday";
  case 5: dayString2 = "Wednesday";
          break;
  default: dayString3 = "Invalid day";
           break;
}
```

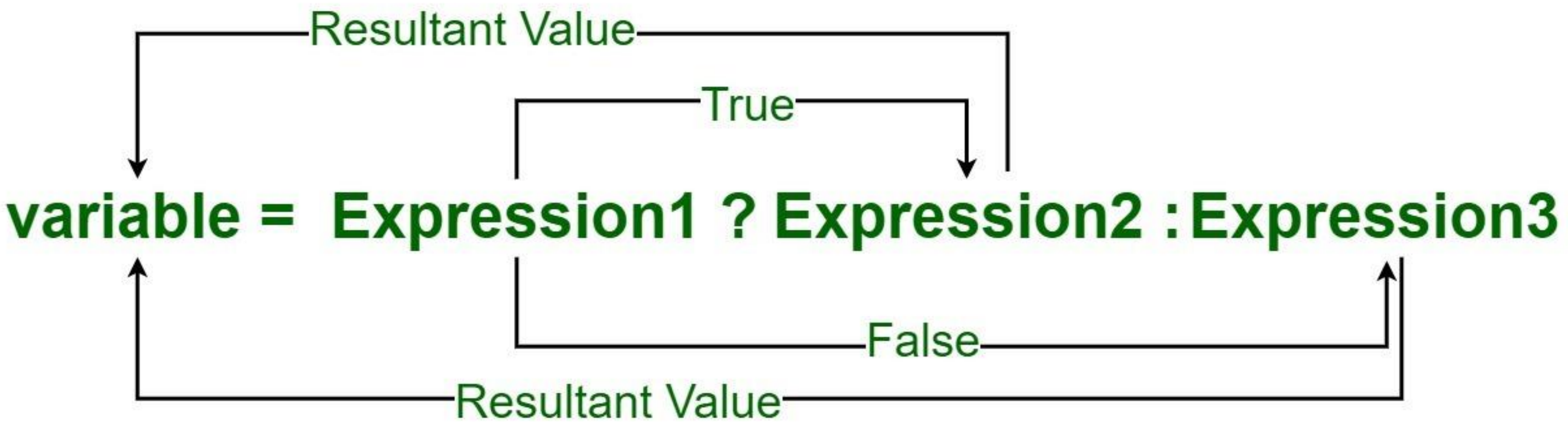
Exercise

- Write a program that will read the value of x and evaluate the following function

$$y = \begin{cases} 1 & \text{for } x > 0 \\ 0 & \text{for } x = 0 \\ -1 & \text{for } x < 0 \end{cases}$$

- Using
  - nested **if** statements,
  - **else if** statements,
  - Conditional and ternary operators **?:**
  - **Read more about conditional operator**
    - https://www.javatpoint.com/conditional-operator-in-java

Conditional or Ternary Operator (?:) in Java



Independent Study - Math class

Read : https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

The package Math has many methods (operations) that you can use to do mathematical operations on numbers:

- Math.max(x,y): returns the highest value of x and y.
- Math.min(x,y): returns the lowest value of x and y.
- Math.sqrt(x): calculates the square root of x.
- Math.abs(x): returns the absolute positive value of x.
- Math.random(): returns a random number between 0.0 (inclusive) and 1.0 (exclusive).
- Math.log(x): returns the logarithm of x

Example:

```
int num = Math.max(2, 8);
System.out.println(num);
```

$\longrightarrow$ Output: 8

You can see all methods available here: https://docs.oracle.com/javase/8/docs/api/java/lang/Math.html

Try out Examples:

https://www.javatpoint.com/java-math

https://www.w3schools.com/java/java_math.asp

Independent Study

- Complete Recommended reading *: **Java for Everyone** - Chapter 02 / Chapter 03*
- Tryout all coding examples provided in lecture slides using code editor and observe output.
- Attempt all exercises provided in lecture slides using code editor and discuss your issues during tutorials.
- Complete Formative test week 02 (Available in Blackboard Week2 folder).
- Attempt all questions in tutorial 01 and submit to BB before deadline
- Solve the following exercises in HackerRank (Will be part of tutorial 02)
  - Java Loops 1&2
  - Java Datatypes
  - Java int to String

Independent Study

- Access to HackerRank Interview Questions
  - https://www.hackerrank.com/domains/java

- Solve following exercises
  - Welcome to Java!
  - Java Stdin and Stdout I
  - Java Stdin and Stdout II
  - Java If-Else
  - Java Datatypes
  - Java int to String

THANK
YOU