Welcome to the

Java Course

Module 1 – Day 03

Content of the course

- Introduction to programming
- Basic programming concepts
- Variables and Data Types
- Conditionals
- **Loops**
- **Control Structures**
- **Introduction to algorithms**

Day 1

Day 2

Day 3

.
.
.

Project Students - Step 2

- If the user enters a birth year in the future, request the birth year again.
- Fix the program to take into account the birth day and month to calculate the student's age.
- Update the date of birth to be displayed using the name of the month.

Project Students - Step 2

```
Enter first name: Ana
Enter last name: Gaggero
Enter birthday (day of month): 22
Enter birth month: 10
Enter birth year: 1982
Enter course registered: Java

Student Name: Ana Gaggero
Date of Birth: 22 October 1982
Age: 41
Course Registered: Java
```

Project Tic Tac Toe - Step 2

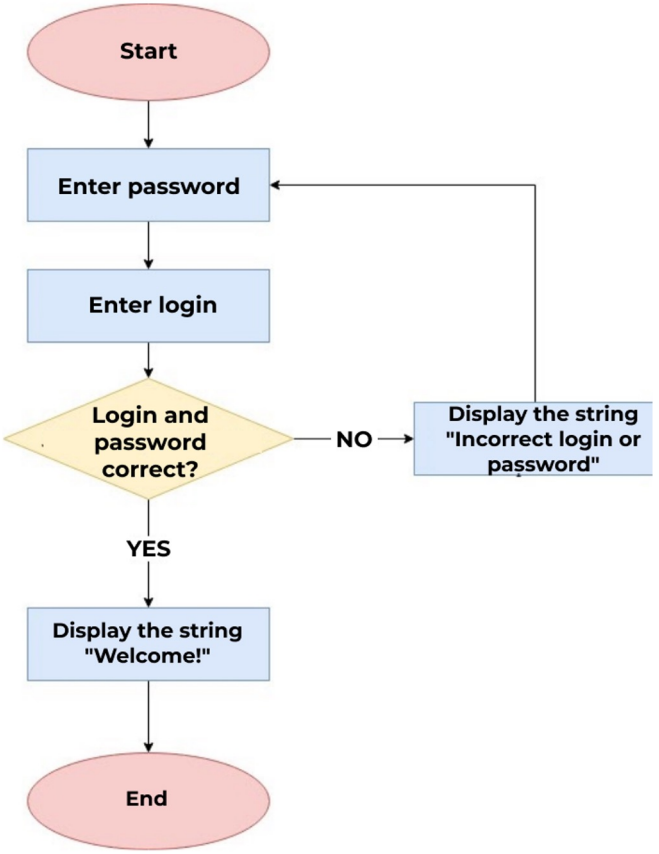
After printing the board, ask the first player to choose a move, then print the board again with the player's choice marked with an X

```
What is your name Player 1? Ana
What is your name Player 2? Juan

Ana will be X and Juan will be O

 1 | 2 | 3
 4 | 5 | 6
 7 | 8 | 9
Ana choose your move: 5
 1 | 2 | 3
 4 | X | 6
 7 | 8 | 9
```

Loop

A loop is a sequence of actions **repeated** a known or unknown number of times.

Loops


Start
Enter password
Enter login
Login and password correct?
NO
Display the string "Incorrect login or password"
YES
Display the string "Welcome!"
End

While Loop

When you don’t know how many times you have to perform a block of code, you need the while loop:

While Loop

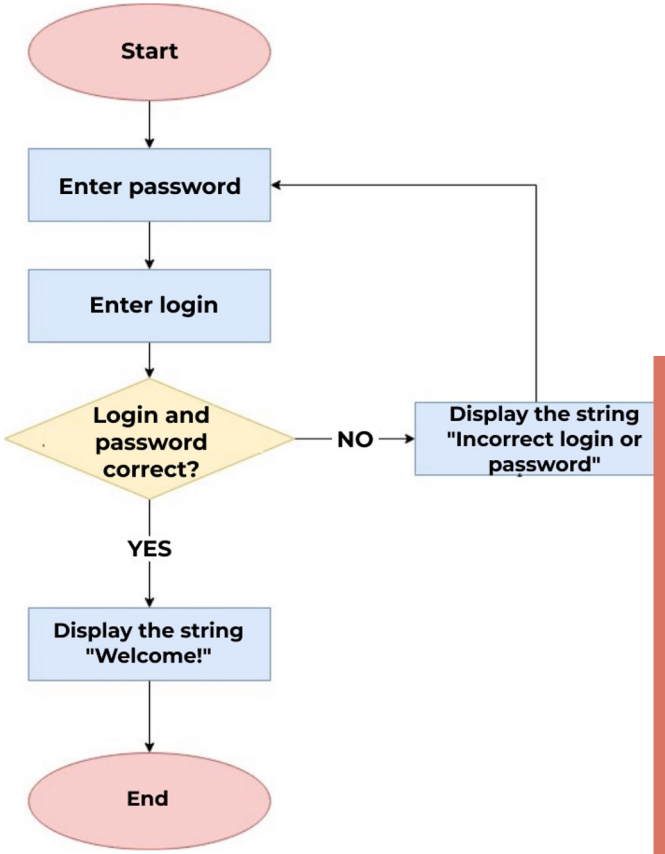
In a while loop, the compiler first checks the condition and then executes the code inside the loop.

While Loops

```
while (repetition condition) {
        Command 1
        Command 2
        ……
        Command n
}
continuation of the program
```

Loop

Command block



While Loop

What happens in this case?

```
1 while (true) {
2 System.out.println(“hello!”);
3 }
```

While Loop

To avoid an infinite loop, the condition **must change** eventually!

Counter

The counter is a variable that stores the number of repetitions of a certain loop.

```
1 int counter = 0;
2 while (counter < 10) {
3   System.out.println(“hello!”);
4   counter++;
5 }
```

Now YOUR TURN !

Let’s do exercises number 1

String operations

String charAt()

Return the char at the given index

```
1 char charAt(int index)
2
3 // usage
4 String s = "Hello";
5
6 char zero = s.charAt(0); // H
7 char one = s.charAt(1); // e
8 char two = s.charAt(2); // l
9 char three = s.charAt(3); // l
10 char four = s.charAt(4); // o
```

String concatenation

Add two string together

```
String s1 = “Hello “;
String s2 = “world !”;

String concat = s1 + s2;

String concat2 = s1.concat(s2);
```

String toLowerCase()

Return the String in lower case

```
String str = "HeLo";
String lowercase = str1.toLowerCase();
```

String toUpperCase()

Return the String in upper case

```
String str = "HeLo";
String uppercase = str1.toUpperCase();
```

String equals()

Compares two strings values

```
String str1 = "HeLo";
String str2 = "Hello";

if (str1.equals(str2)
  System.out.println(str1 + " equals " + str2);
else
  System.out.println(str1 + " is not equal to " + str2);

// the result is false
```

String equalsIgnoreCase()

Compares two strings values ignoring upper or lower case

```
String str1 = "HeLo";
String str2 = "Hello";

if (str1.equalsIgnoreCase(str2)
  System.out.println(str1 + " equals " + str2);
else
  System.out.println(str1 + " is not equal to " + str2);
```

String contains()

Checks if a String contains another

```
String str = "HeLo";

if (str.contains("H")
  System.out.println(str + " contains H");
else
  System.out.println(str + " does not contain H ");

// The result is true
```

String startsWith()

Checks if a String starts with a substring

```
String str = "HeLlo";

if (str.startsWith("H"))
System.out.println(str + " starts with H");
else
System.out.println(str + " does not start with H");
```

String endsWith()

Checks if a String ends with a substring

```
String str = "HeLlo";

if (str.endsWith("H"))
System.out.println(str + " ends with H");
else
System.out.println(str + " does not end with H");
```

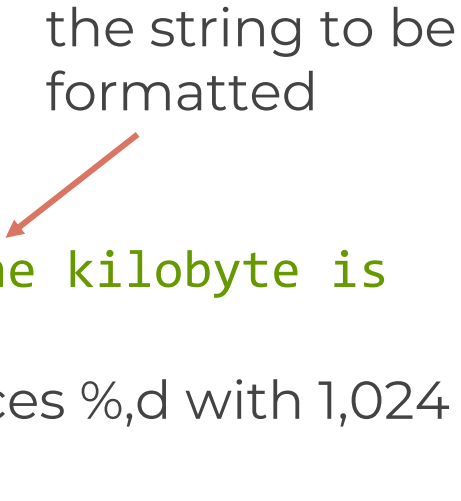
String.format()

When more complex formatting is needed, we can use the *format* function from the *String* class.

This function will replace tokens that start with % by the corresponding arguments passed to the function.

Example:

```
String myStr = String.format("Hello %s! One kilobyte is
%,d bytes.", "World", 1024);
```



String.format()

are the most common ones:

%s — string (anything; calls String.valueOf)
%d — integer (byte/short/int/long, or their wrappers)
%f — floating-point (float/double)
%b — boolean
%c — character
%e — scientific notation (floats)
%tX — date/time (e.g., %tY year, %tm month, %td day)
%% — a literal percent sign
%n — newline (platform-independent)

Now YOUR TURN !

Let’s do exercises number 2

Type casting

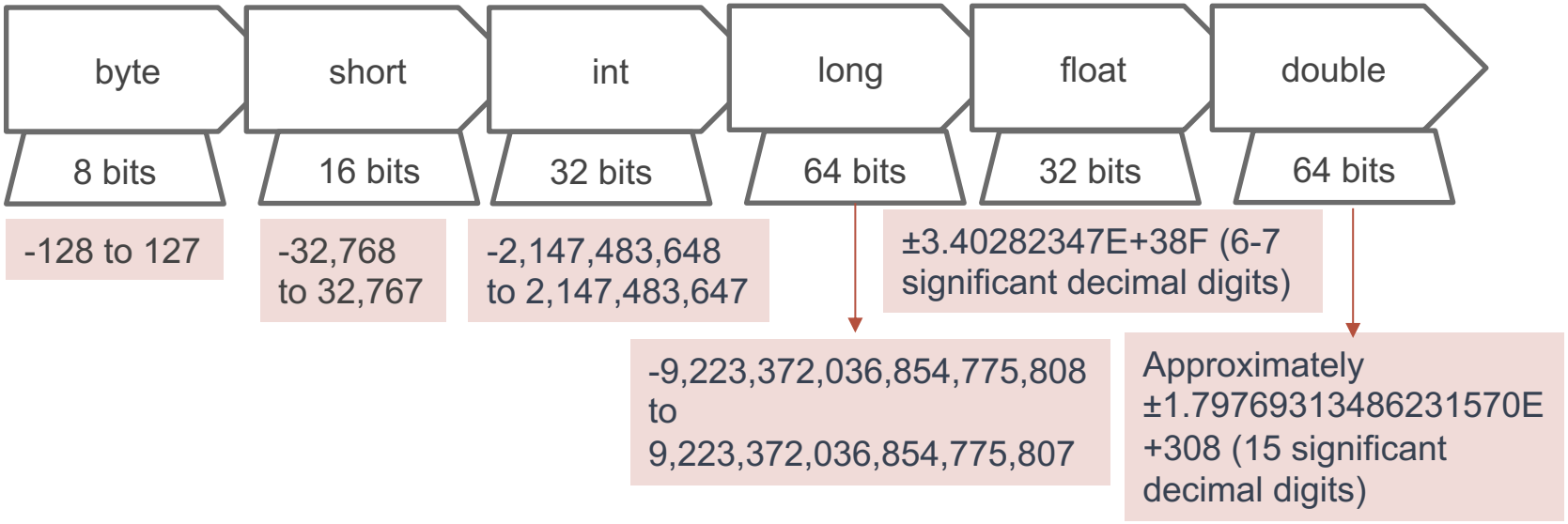
Convert variables from one type to another:

- **Implicit**: Safe.
- **Explicit**: Must be done manually, risk of data lost.

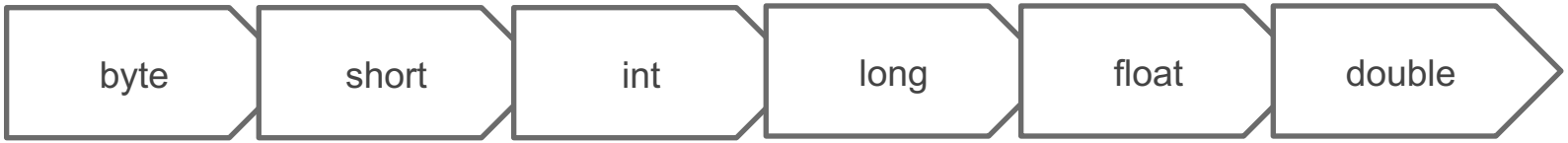
Type casting

Implicit

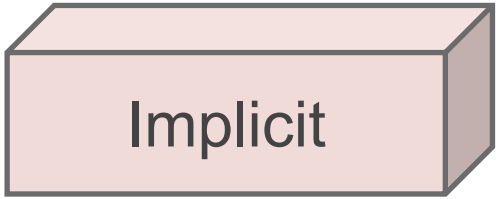
```
1 int myInt = 9;
2 long myLong = myInt;
```



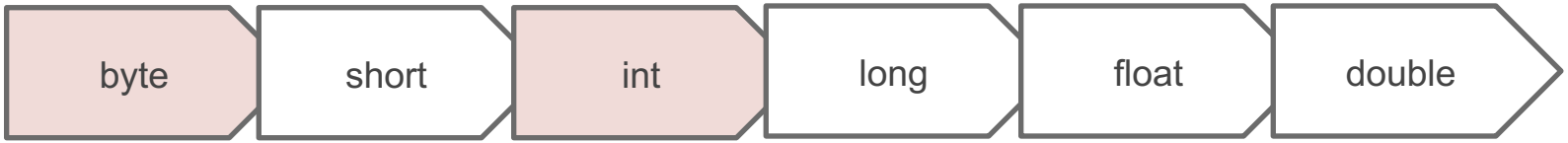
Type casting




```
1 byte myByte = 100;
2 int myInt = myByte;
```

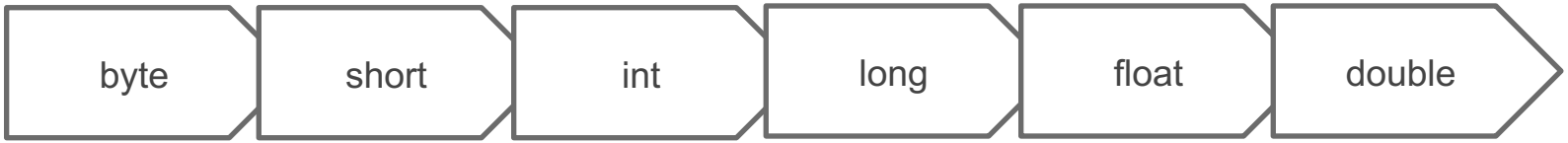


Type casting

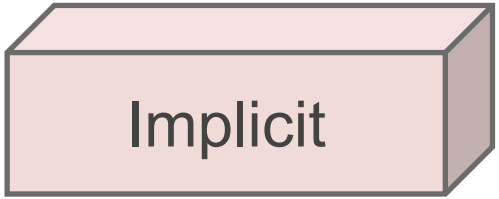



```
1 byte myByte = 100;
2 int myInt = myByte;
```

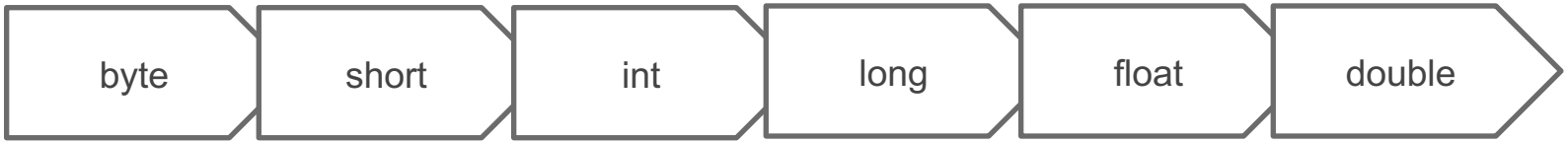
Type casting




```
1 double myDouble = 9.78;
2 int myInt = myDouble;
```



Type casting



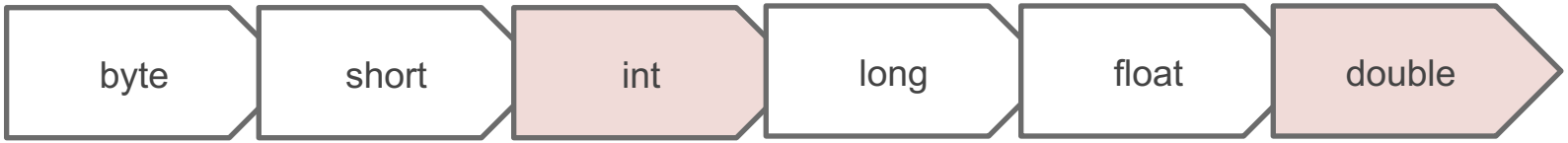

```
1 double myDouble = 9.78;
2 int myInt = myDouble;
```

Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Type mismatch: cannot convert from double to int

Implicit

Explicit

Type casting




```
1 double myDouble = 9.78; // Outputs 9.78
2 int myInt = (int) myDouble; // Outputs 9
```

Now YOUR TURN !

Let’s do exercises number 3

Example: conversion from short to byte

```
short shortVal = 128;
byte byteVal = (byte) shortVal;
```

The value of byteVal will be **-128**

2’s complement

- Start with the absolute binary representation of the number.
- Invert (or flip) all bits – changing every 0 to 1, and every 1 to 0.
- Add 1 to the entire inverted number, ignoring any overflow.

Example: get -6 in 2’s complement

Step 1: the binary representation of 6 is 0110

Step 2: after inverting all bits we get 1001

Step 3: 1001 + 1 in binary is 1010

Project Students - Step 3

Modify the program such that it asks for the birth year until the user enters a valid one

```
Enter first name: Ana
Enter last name: Gaggero
Enter birthday (day of month): 22
Enter birth month: 10
Enter birth year: 2027
The birth year cannot be in the future, please enter a valid birth year: 2028
The birth year cannot be in the future, please enter a valid birth year: 1982
Enter course registered: Java

Student Name: Ana Gaggero
Date of Birth: 22 October 1982
Age: 41
Course Registered: Java
```

Project Tic Tac Toe

Step 3

- Add a loop to allow 9 moves.
- Add a variable to keep track of who's turn it is
- Check if the place is empty before updating the board. If it's not empty, print "Invalid move"

```
Ana will be X and Juan will be O
 1 | 2 | 3
 4 | 5 | 6
 7 | 8 | 9

Ana choose your move: 5
 1 | 2 | 3
 4 | X | 6
 7 | 8 | 9

Juan choose your move: 1
 O | 2 | 3
 4 | X | 6
 7 | 8 | 9

Ana choose your move: 1
Invalid move.
Ana choose your move: 3
 O | 2 | X
 4 | X | 6
 7 | 8 | 9
```