Programming Fundamentals

Conditions in Java

Dr Ahmed Al-Azawei

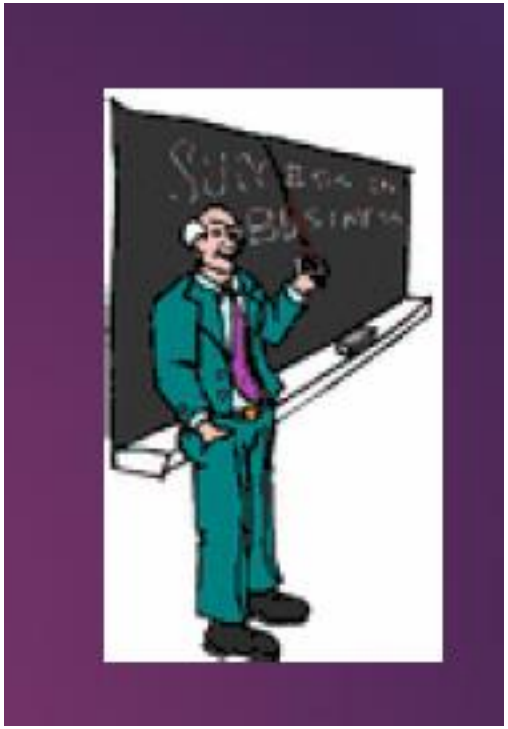
University of Babylon

2022/2023

Lecture 05

Outline

- Decisions, Decisions, Decisions
- If conditions
  - If statement
  - If else statement
  - If else if statement
- Switch conditions



Background

- Previous problem-solving solutions have the straight-line property
  - They execute the same statements for every run of the program

```
public class JavaFun
{
  // main(): application entry point
        public static void main(String[] args)
        {
                System.out.println("I think Java programming is fun");
                System.out.println(" I like learning Java ");
                System.out.print("Java is ");
                System.out.print(" a high level programming ");
                System.out.print("language.");
        }
}
```

Conditional constructs

- Provide
  - Ability to control whether a statement list is executed or not
- Two constructs
  - If statement
    - if
    - if-else
    - if-else-if
  - Switch statement

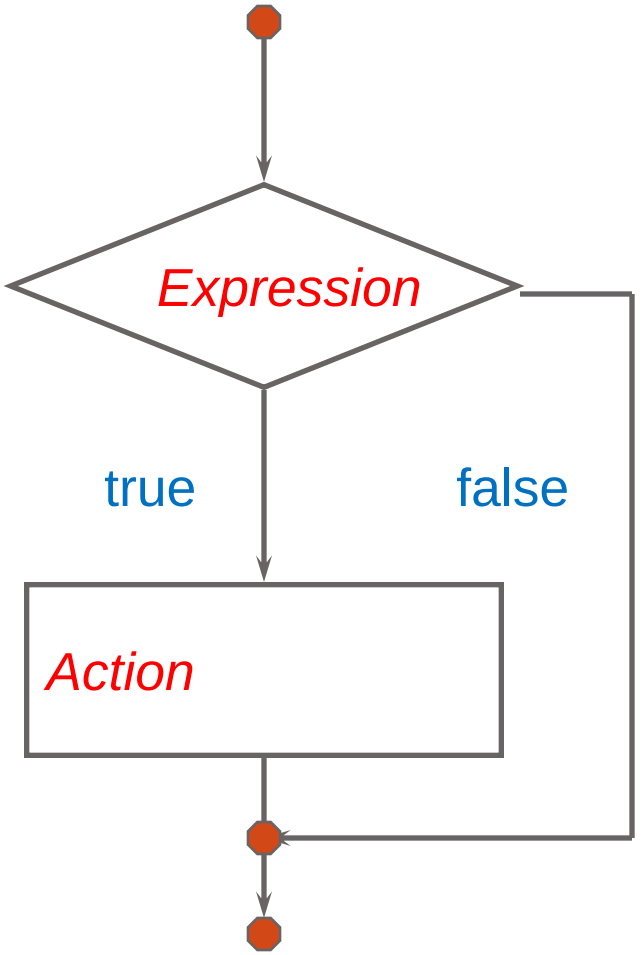
Basic if statement

- Syntax

```
if (Expression)
    Action
```

- If the *Expression* is true then execute *Action*

- *Action* is either a single statement or a group of statements within **braces**



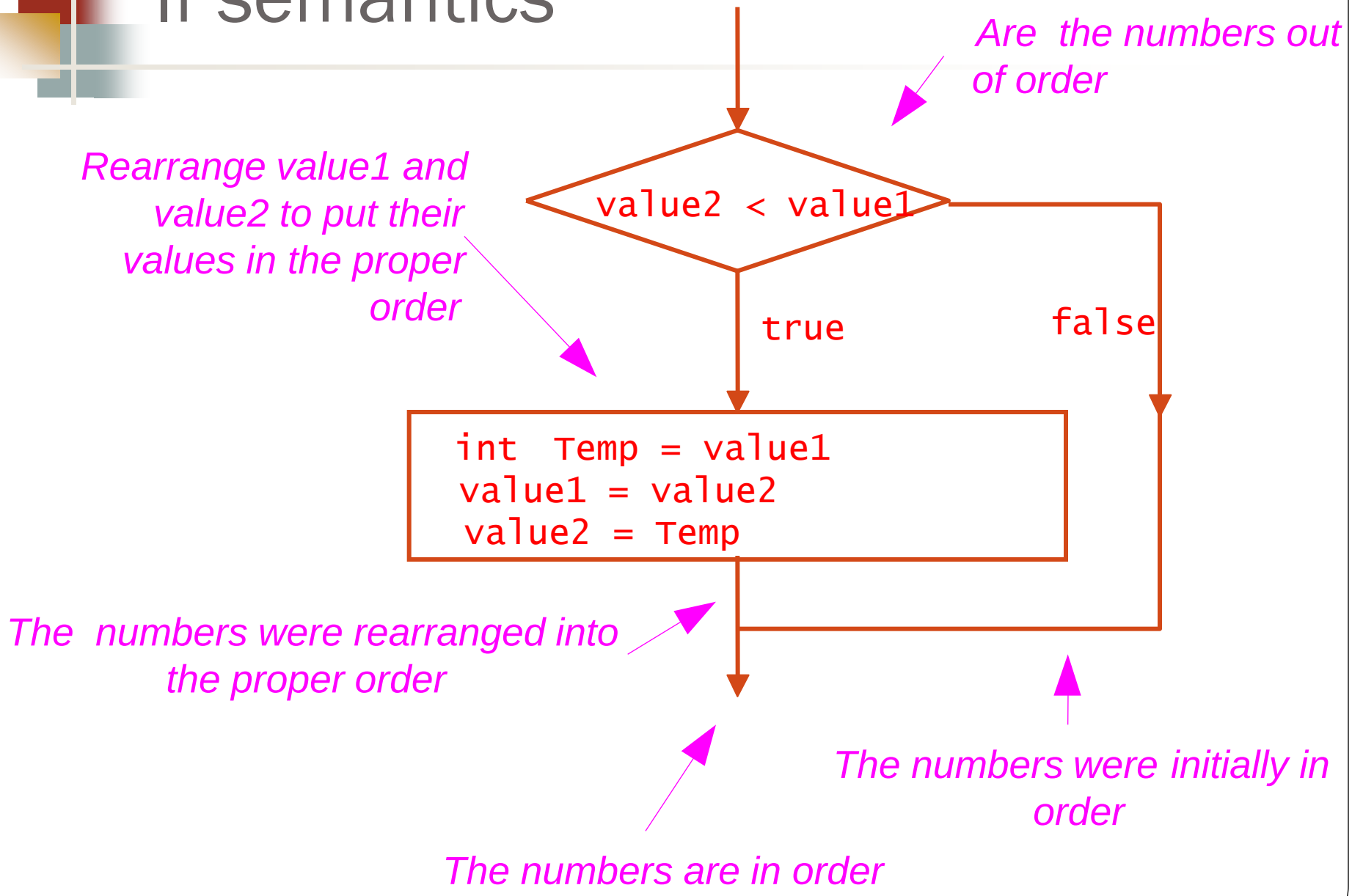
Sorting two values

```
System.out.print("Enter an integer number: ");
int value1 = stdin.nextInt();
System.out.print("Enter another integer number: ");
int value2 = stdin.nextInt();

// rearrange numbers if necessary
if (value2 < value1) {
   // values are not in sorted order
   int Temp = value1;
   value1 = value2;
   value2 = Temp;
}

// display values
System.out.println("The numbers in sorted order are "
   + value1 + " and then " + value2);
```

What happens if the user enters 22 and 28?

What happens if the user enters 22 and 8?

If semantics



What an if statement executes

- An if statement executes the next *block* of code
- **A block is either:**
  - **A single statement** without curly brackets:

```
if (a == b)
  System.out.println (“a==b!!!”);
```

  - **A number of statements** enclosed by curly brackets:

```
if (a == b) {
  System.out.print (“a”);
  System.out.print (“==”);
  System.out.print (“b”);
  System.out.println (“!!!”);
}
```

Why we always use braces

- What is the output?

```
int m = 5;
int n = 10;

if (m < n)
  ++m;
  ++n;

System.out.println(“ m = “ + m + “ n = “ + n);
```

The output is:
m = 6 n = 11

The if-else statement

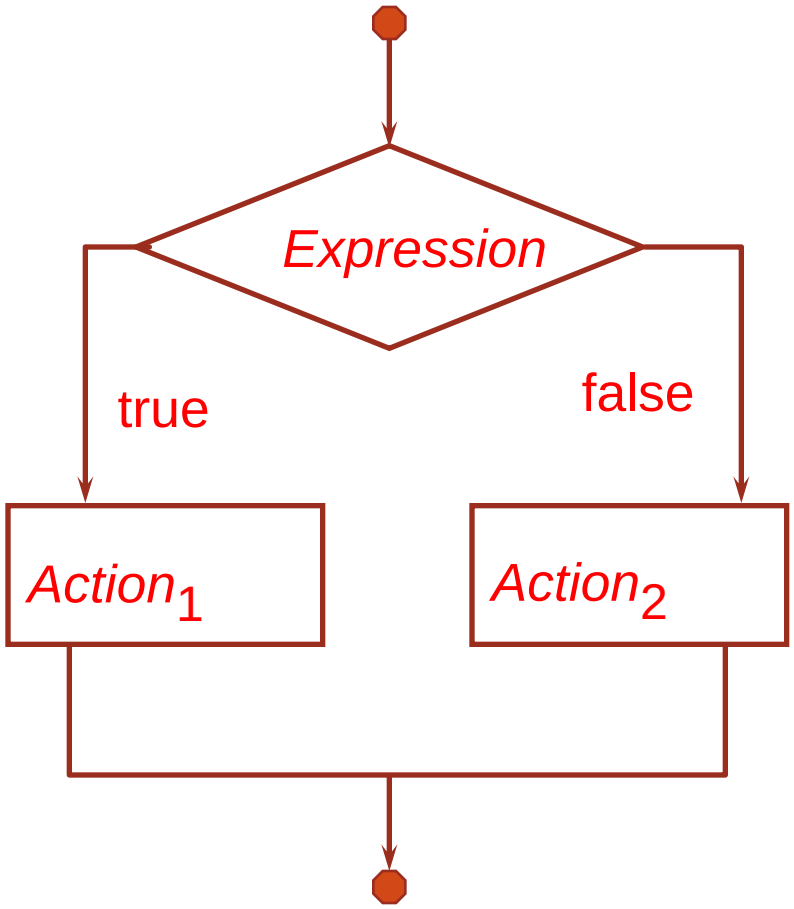
- Syntax

`if` (*Expression*)

Action$_1$

`else`

Action$_2$



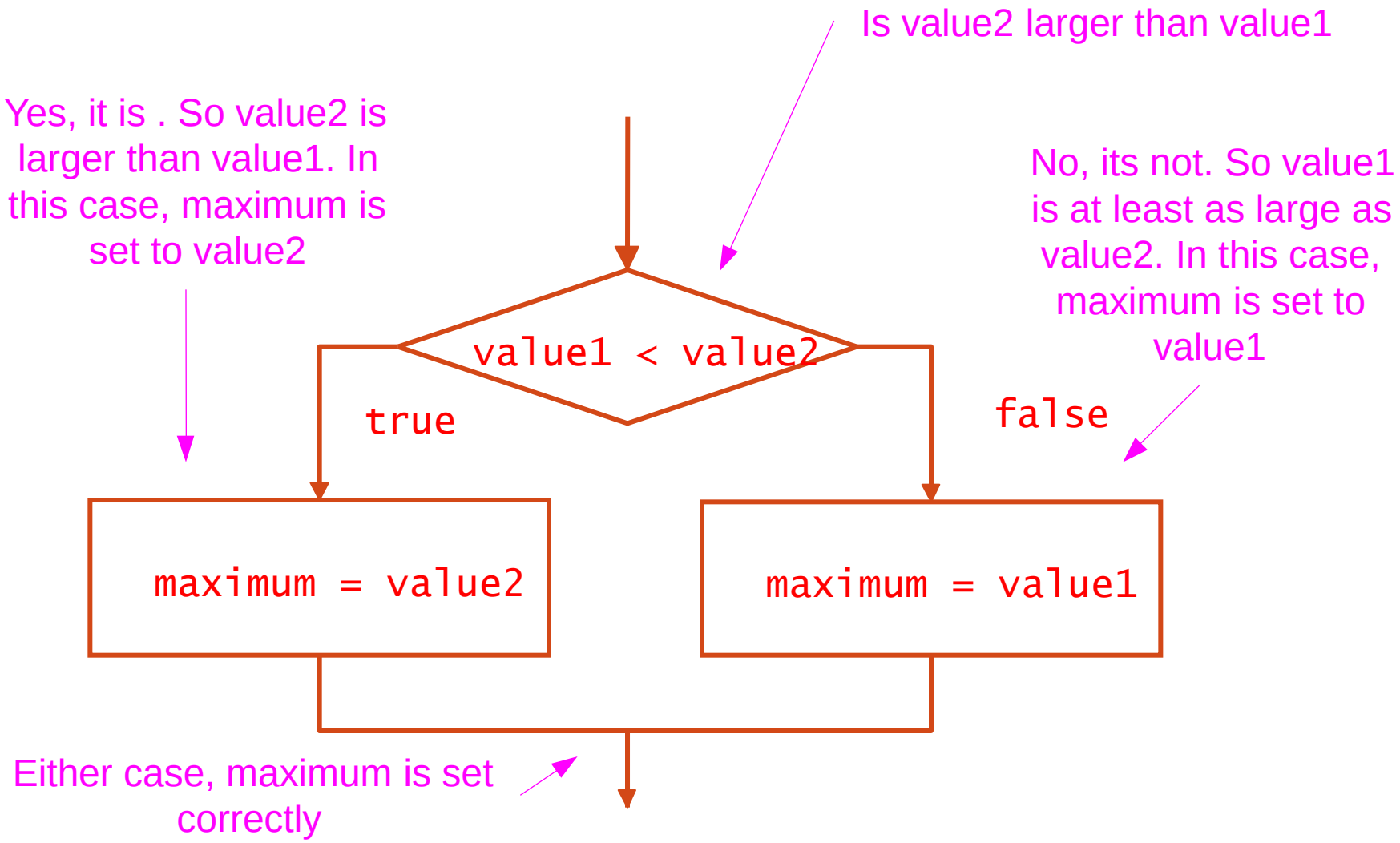
- If *Expression* is true then execute *Action*$_1$ otherwise execute *Action*$_2$
- The actions are either a single statement or a list of statements within braces

Finding the maximum of two values

```
System.out.print("Enter an integer number: ");
int value1 = stdin.nextInt();
System.out.print("Enter another integer number: ");
int value2 = stdin.nextInt();

int maximum;
if (value1 < value2) {    // is value2 larger?
   maximum = value2;      // yes: value2 is larger
}
else { // (value1 >= value2)
   maximum = value1;      // no: value2 is not larger
}
System.out.println("The maximum of " + value1
        + " and " + value2 + " is " + maximum);
```

But is it initialized?

Finding the maximum of two values



How do you like your braces?

```
if (a == b)
{
   //...
}
else {
   //...
}
```

```
if (a == b)
{
   //...
} else
{
   //...
}
```

```
if (a == b)
{
   //...
}
else
{
   //...
}
```

```
if (a == b) {
   //...
}
else {
   //...
}
```

```
if (a == b) {
   //...
} else {
   //...
}
```

If-then-else precedence

```
if (number != 0)
    if (number > 0)
        System.out.println("positive");
    else

    System.out.println("negative");
```

Which if does this else refer to?

If-else-if Statement

- Consider

We can change the whitespace of the code

```
if (number == 0) {
   System.out.println("zero");
}
else {
     if (number > 0) {
        System.out.println("positive");
     }
     else {
        System.out.println("negative");
     }
 }
```

These braces aren't needed

Same results as previous segment – but this segment better expresses the meaning of what is going on

Sorting three values

- For sorting values n1, n2, and n3 there are six possible orderings
  - $n1 \leq n2 \leq n3$
  - $n1 \leq n3 \leq n2$
  - $n2 \leq n1 \leq n3$
  - $n2 \leq n3 \leq n1$
  - $n3 \leq n1 \leq n2$
  - $n3 \leq n2 \leq n1$

- Suppose s1, s2, s3 are to be a sorted version of n1, n2, and n3

Sorting three values

```
if ((n1 <= n2) && (n2 <= n3)) {       // n1 <= n2 <= n3
  s1 = n1;   s2 = n2;   s3 = n3;
}
else if ((n1 <= n3) && (n3 <= n2)) { // n1 <= n3 <= n2
  s1 = n1;   s2 = n3;   s3 = n2;
}
else if ((n2 <= n1) && (n1 <= n3)) { // n2 <= n1 <= n3
  s1 = n2;   s2 = n1;   s3 = n3;
}
else if ((n2 <= n3) && (n3 <= n1)) { // n2 <= n3 <= n1
  s1 = n2;   s2 = n3;   s3 = n1;
}
else if ((n3 <= n1) && (n1 <= n2)) { // n3 <= n1 <= n2
  s1 = n3;   s2 = n1;   s3 = n2;
}
else { // n3 <= n2 <= n1
  s1 = n3;   s2 = n2;   s3 = n1;
}
```

Finding the minimum value

- Consider:

```
// z is to hold the minimum of x and y
if ( x < y )
  z = x;
else
  z = y;
```

Notice no braces!

- Another way to do this:

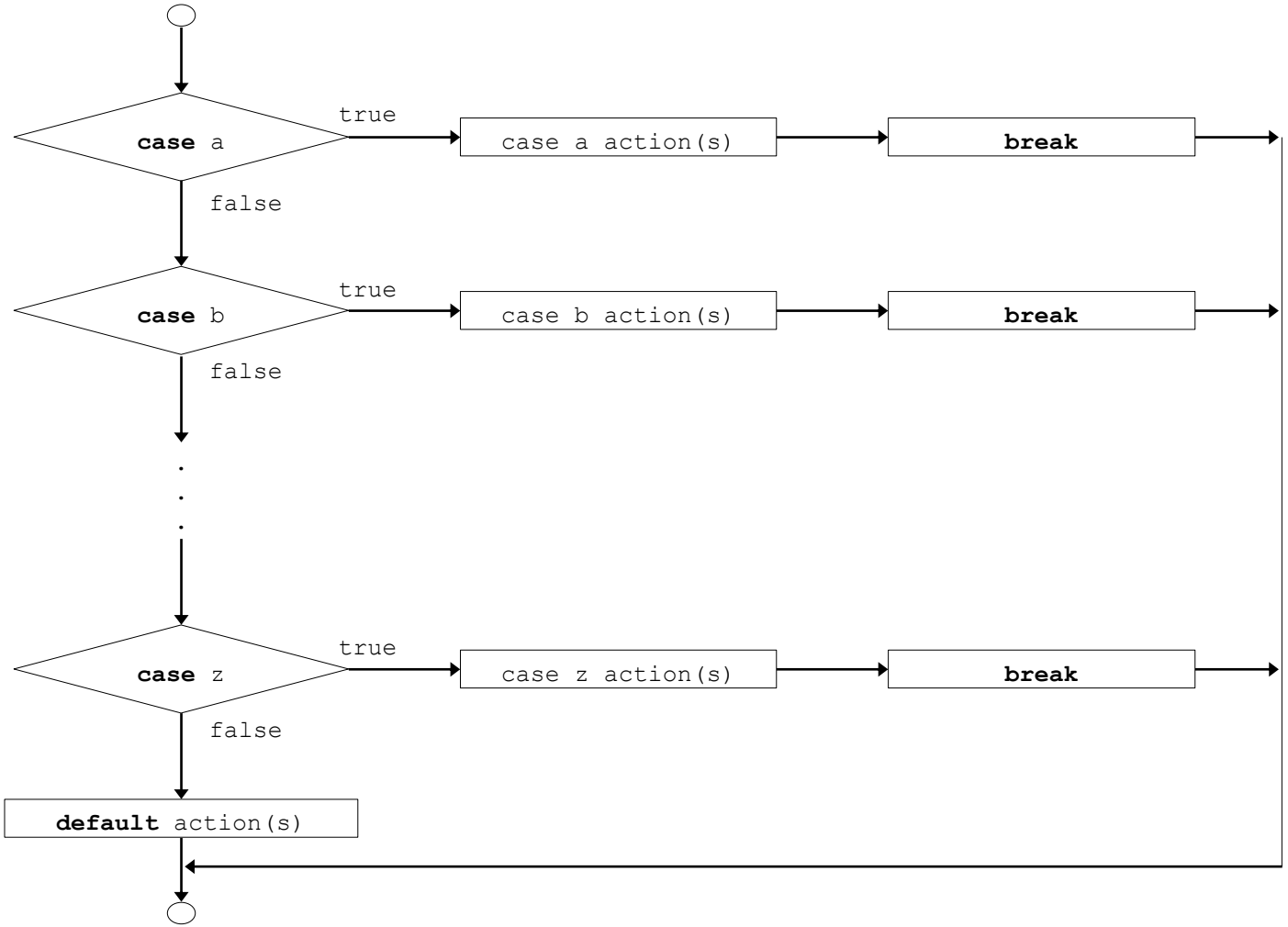
```
z = (x<y) ? x : y;
```

Switch statement

- **switch**
  - Test variable for multiple values
  - Series of **case** labels and optional **default** case

```
switch ( variable ) {
      case value1:         //taken if variable == value1
      statements
      break;               //necessary to exit switch

      case value2:
      case value3:       //taken if variable == value2 or == value3
      statements
      break;

      default:           //taken if variable matches no other cases
      statements
       break;
}
```

Switch statement



A switch statement example

```
if (a == '0')
    System.out.println ("zero");
else if (a == '1')
    System.out.println ("one");
else if (a == '2')
    System.out.println ("two");
else if (a == '3')
    System.out.println ("three");
else if (a == '4')
    System.out.println ("four");
else
    System.out.println ("five+");
```

```
switch (a) {
        case '0':
                System.out.println ("zero");
                break;
        case '1':
                System.out.println ("one");
                break;
        case '2':
                System.out.println ("two");
                break;
        case '3':
                System.out.println ("three");
                break;
        case '4':
                System.out.println ("four");
                break;
        default:
                System.out.println ("five+");
                break;
}
```

A switch statement example

```
import java.util.Scanner;
public class JavaFun {
   // main(): application entry
point
public static void main(String[]
args) {
Scanner stdin=new Scanner
(System.in);
System.out.println("Enter a
value: ");
char s1 = stdin.next().charAt(0);
switch (s1) {
case '0':
System.out.println ("zero");
break;
case '1':
System.out.println ("one");
break;
case '2':
System.out.println ("two");
break;
case '3':
System.out.println ("three");
break;
case '4':
System.out.println ("four");
break;
default:
System.out.println ("five+");
break;
}
}
}
```

Processing a request

```
System.out.print("Enter a number: ");
int n1 = stdin.nextInt();

System.out.print("Enter another number: ");
int n2 = stdin.nextInt();

System.out.print("Enter desired operator: ");
char operator = stdin.nextLine().charAt(0);

switch (operator) {
   case '+' : System.out.println((n1 + n2)); break;
   case '-' : System.out.println(n1 - n2); break;
   case '*' : System.out.println(n1 * n2); break;
   case '/' : System.out.println(n1 / n2); break;
   default: System.out.println("Illegal request");
}
```

charAt () method

The **Java String charAt(int index)** method returns the character at the specified index in a string. The index value that we pass in this method should be between 0 and (length of string-1). For example: s.charAt(0) would return the first character of the string represented by instance s.

```
public class CharAtExample {
  public static void main(String args[]) {
                String str = "Welcome to string handling tutorial";
                //This will return the first char of the string
                char ch1 = str.charAt(0);
                //This will return the 6th char of the string
                char ch2 = str.charAt(5);
                //This will return the 12th char of the string
                char ch3 = str.charAt(11);
                //This will return the 21st char of the string
                char ch4 = str.charAt(20);
                System.out.println("Character at 0 index is: "+ch1);
                System.out.println("Character at 5th index is: "+ch2);
                System.out.println("Character at 11th index is: "+ch3);
                System.out.println("Character at 20th index is: "+ch4);
  }
}
```

Testing variables for equality

- Consider

```
System.out.print("Enter an integer number: ");
int n1 = stdin.nextInt();
System.out.print("Enter another integer number: ");
int n2 = stdin.nextInt();

if (n1 == n2) {
  System.out.println("Same");
}
else {
  System.out.println(“Different");
}
```

What is the output if the user enters 88 and 3?

What is the output if the user enters 88 both times?

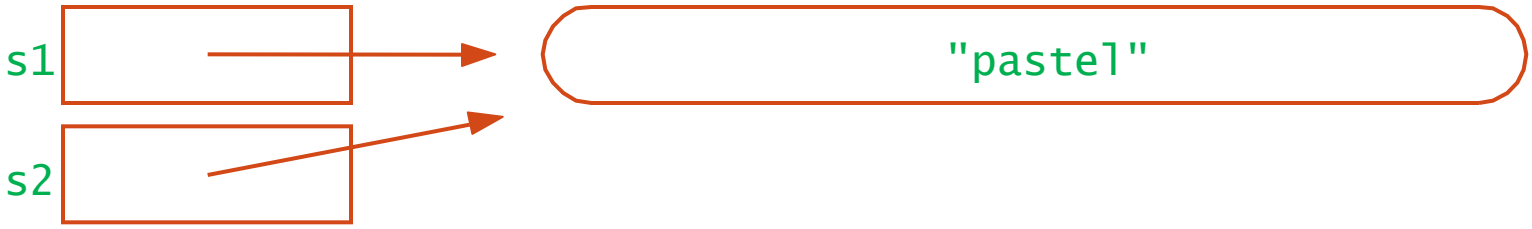
Testing objects for equality

- Consider

```
String s1 = “pastel”;
String s2 = s1;

if (s1 == s2) {
  System.out.println("Same");
}
else {
  System.out.println("Different");
}
```

Testing objects for equality

- Memory looks like



- The comparison is between the references!

- Thus, s1 and s2 are the same (they refer to the same object)

Testing objects for equality

Consider:

```
System.out.print("Enter a string: ");
String s1 = stdin.nextLine();
System.out.print("Enter another string: ");
String s2 = stdin.nextLine();

if (s1 == s2) {
  System.out.println("Same");
}
else {
  System.out.println("Different");
}
```

What is the output if the user enters "pastel" both times?

Testing objects for equality

- When it is executed

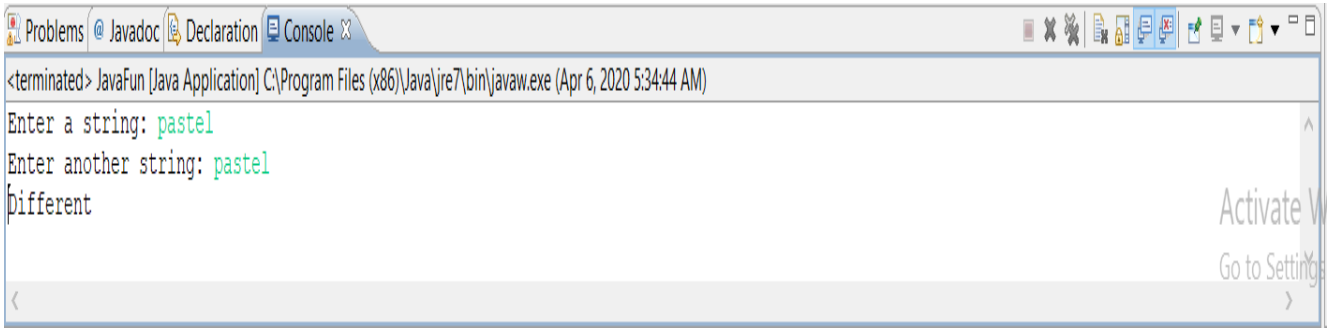
```
System.out.print("Enter a string: ");
String s1 = stdin.nextLine();
System.out.print("Enter another string: ");
String s2 = stdin.nextLine();
```

- Memory looks like

- As a result no matter what is entered s1 and s2 are not the same
  - They refer to different objects

Testing objects for equality

```
import java.util.Scanner;
public class JavaFun {
   // main(): application entry point
    public static void main(String[] args) {
        Scanner stdin=new Scanner (System.in);
        System.out.print("Enter a string: ");
        String s1 = stdin.nextLine();
        System.out.print("Enter another string: ");
        String s2 = stdin.nextLine();

        if (s1 == s2) {
            System.out.println("Same");
        }
        else {
            System.out.println("Different");
        }


}
}
```



Testing operators for equality

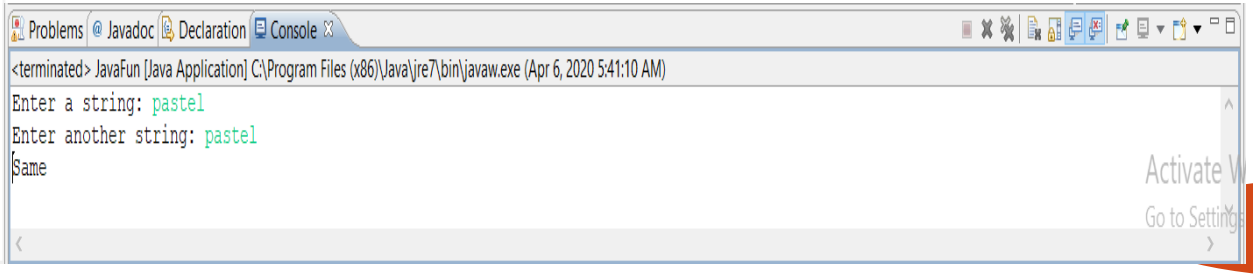
```
System.out.print("Enter a string: ");
String s1 = stdin.nextLine();
System.out.print("Enter another string: ");
String s2 = stdin.nextLine();

if (s1.equals(s2)) {
  System.out.println("Same");
}
else {
  System.out.println("Different");
}
```

Tests whether s1 and s2 represent the same object

Most classes have a method equals(). It compares the objects themselves, not the references.

Testing operators for equality

```
import java.util.Scanner;
public class JavaFun {
   // main(): application entry point
    public static void main(String[] args) {
        Scanner stdin=new Scanner (System.in);
        System.out.print("Enter a string: ");
        String s1 = stdin.nextLine();
        System.out.print("Enter another string: ");
        String s2 = stdin.nextLine();

        if  (s1.equals(s2))  {
            System.out.println("Same");
        }
        else {
            System.out.println("Different");
        }


}
}
```



thank you!