Object-Oriented Programming I

Branching the Program Flow

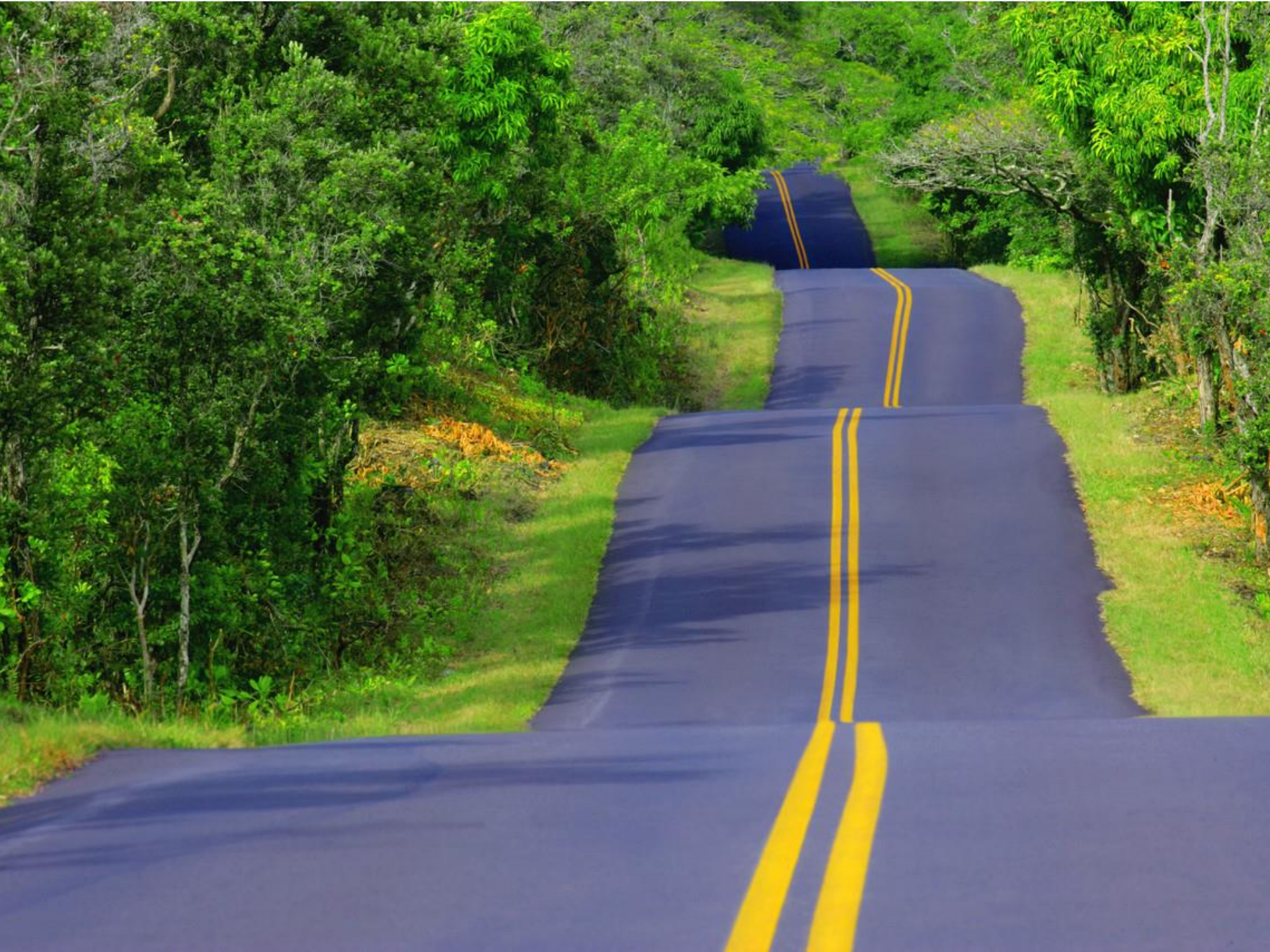
Slides by Magdin Stoica
Updates by Georg Feil

Learning Outcomes

1. Explain the need for branching a sequence of statements in computer programs
2. Define control structures using if-else statement
3. Explain how if-else statements control the flow of the program
4. Create programs that using flow control structures to achieve different outcomes in the program based on program conditions

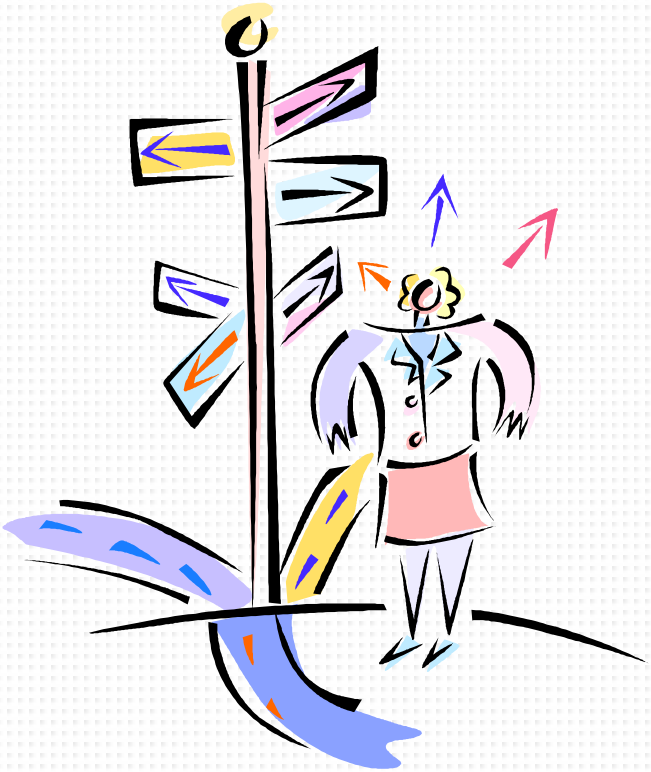
Reading Assignment

- Introduction to Java Programming (required)
  - Chapter 3: Selections, sections 3.1 to 3.10
    - Except 3.8 (we'll be reading that in a week or two)



Conditions and branching

- *Conditional branching is what makes a computer seem "smart"*
- While playing a game the user may, depending on a condition,
  - Win
  - Loose
- When the user enters input it may, depending on a condition,
  - Print a message, "Input OK"
  - Print a different message, "Error"
- When the user makes a request it may be, depending on a condition,
  - Approved
  - Denied



A fork in the road: Banking

Positive outcome: what happens if the condition is “true”

…

Account has sufficient funds?

1. Balance decreases
2. Money dispensed

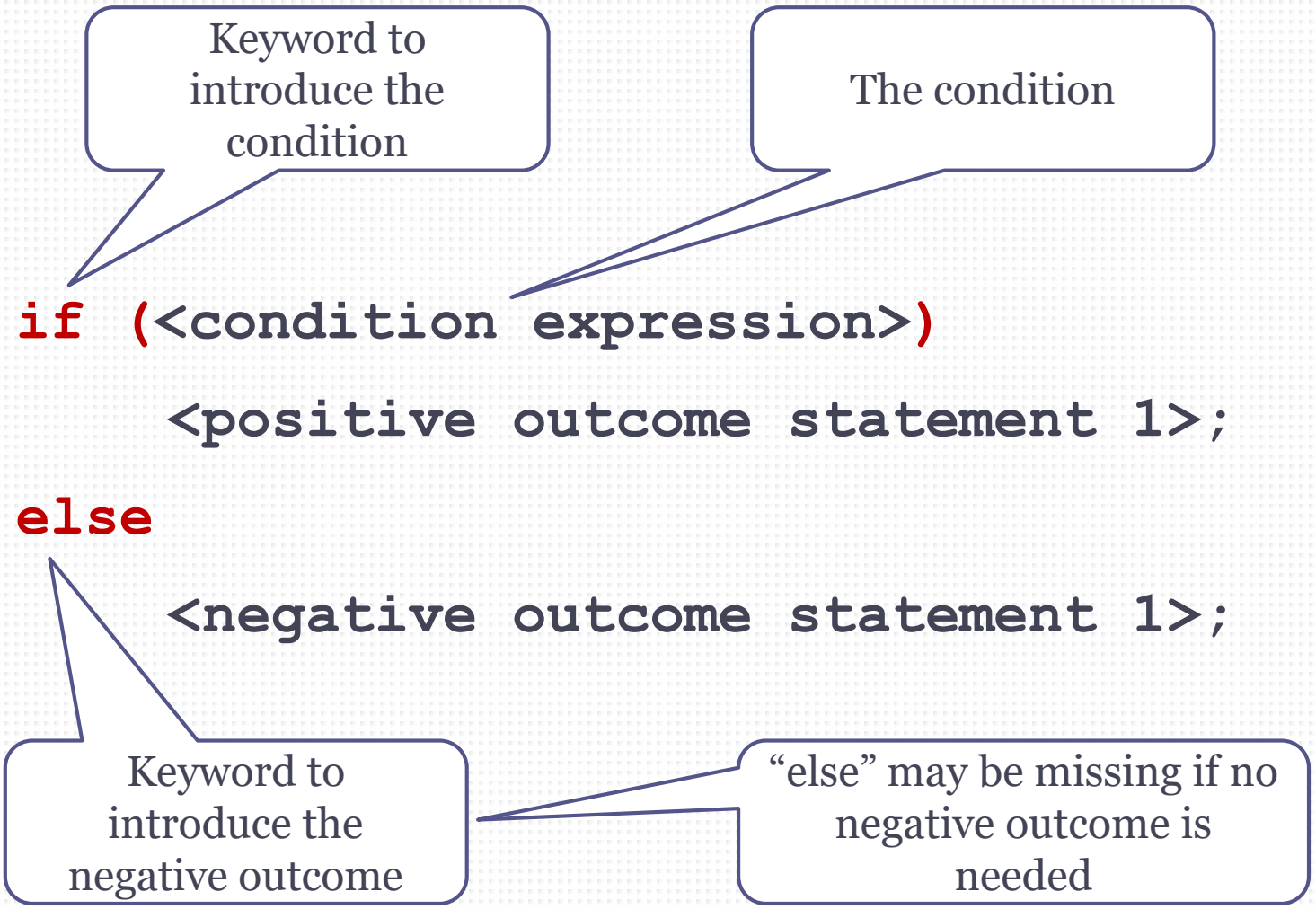
1. User is informed of insufficient funds

…

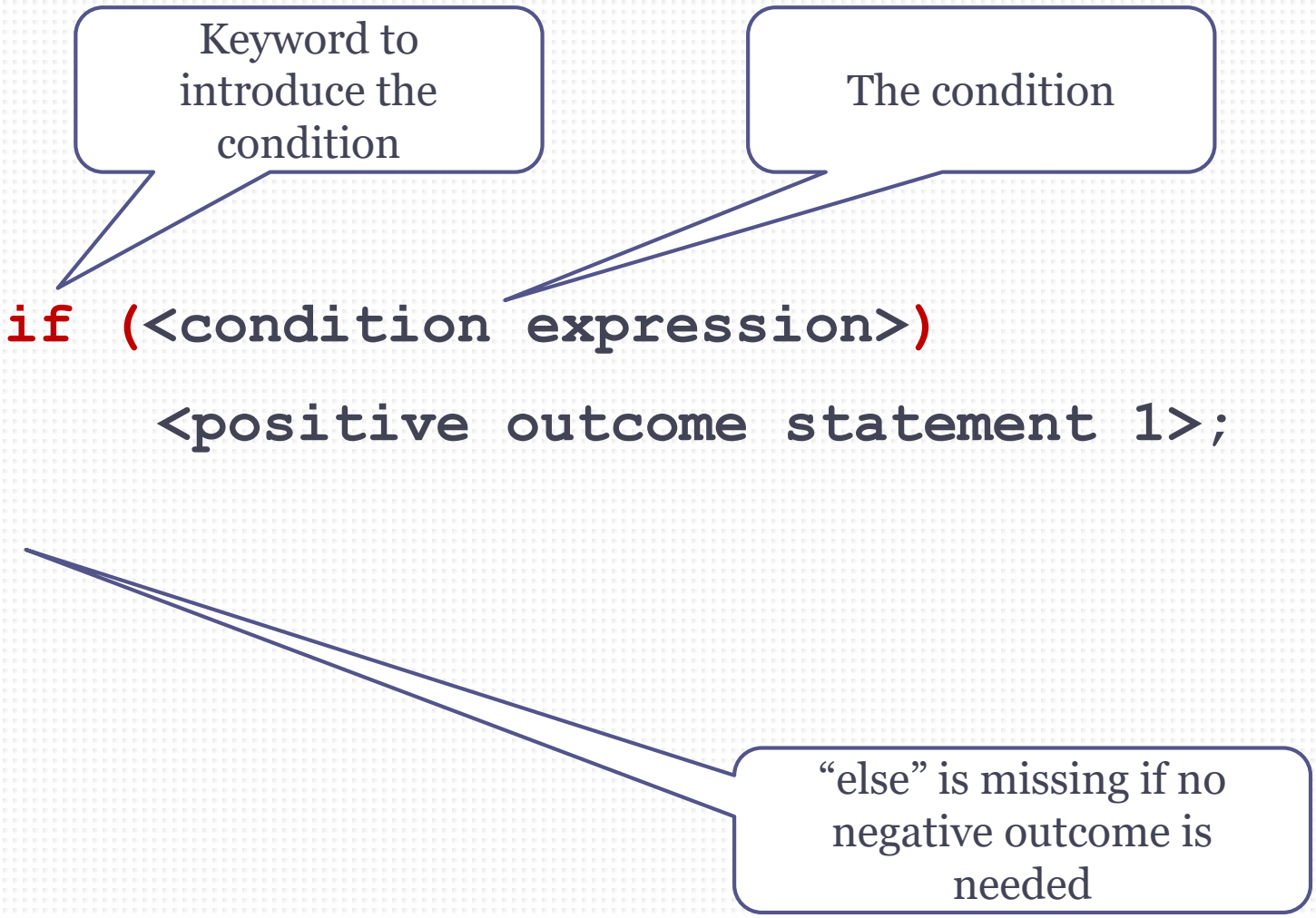
The condition

Negative outcome: what happens if the condition is “false”

If / Else Control Structure (pseudocode)



If Statement (without an else)



If / Else Statement in Practice

The "if block".

Always use curly brackets even if only one statement is needed

The "else block".

Always use a curly brackets even if only one statement is needed

```
if (grade >= 80) {
    System.out.println("Grade A!");
    System.out.println("Well done");
}
else {
    System.out.println("B+ or less");
    System.out.println("Study more?");
}
```

Be Proactive!

Always use { } blocks with all control statements

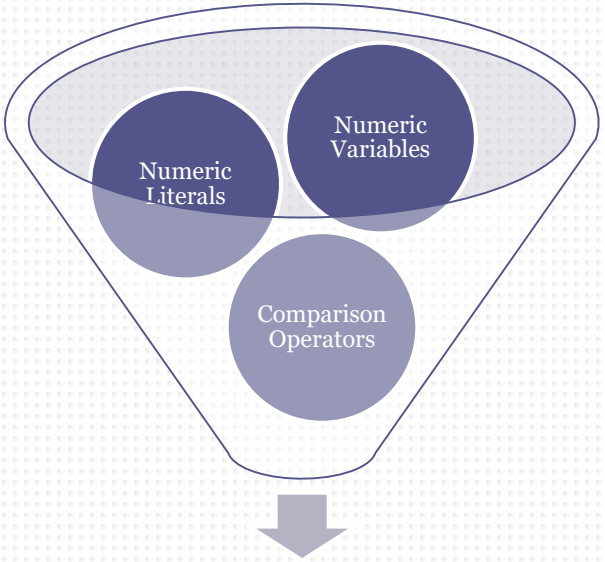
Don’t use “else” unless you need it

-

(Avoid empty blocks)

Comparison Operators

- Two values of a related type (e.g. two numbers, two characters) can be compared using comparison operators
- The result of using a comparison operator is always a boolean value, true or false
  - `<` is the "less than" operator.
  - `<=` is the "less than or equal to" operator.
  - `==` is the "equal to" operator.
  - `!=` is the "not equal to" operator.
  - `>` is the "greater than" operator.
  - `>=` is the "greater than or equal to" operator



Comparison Operators (cont.)

- < The resulting value is true if the left operand is less than the right operand and false otherwise
- <= The resulting value is true if the left operand is less than OR equal to the right operand and false otherwise
- == The resulting value is true if the left operand has the same value as the right operand and false otherwise.
- != The resulting value is true if the left operand is NOT the same value as the right operand and false otherwise.
- > The resulting value is true if the left operand is greater than the right operand and false otherwise
- >= The resulting value is true if the left operand is greater than OR equal to the right operand and false otherwise

If / Else Common Errors

```
if (<condition expression>);
{
    <positive outcome statement 1>;
    <positive outcome statement 2>;
    …
}
```

Logic error (compilers will let this go) but the positive outcome statement will always execute regardless of the condition expression

If / Else Common Errors

```
if (number = 77) {
    …
}
```

Syntax error. This is the assignment operator. The expression value is a number not a Boolean.

Use == instead to test for equality

The IF statement does not require a ;

The IF statement uses

==

for equality checks

not the

=

Statements inside if/else blocks

- The statements inside the if or the else blocks can be any statements
- Declaring variables inside if and else blocks
  - Variables declared inside if/else blocks are local to those blocks
  - A variable declared inside the "if" block is not available in the else block nor outside of the if/else control structure
  - A variable declared inside the "else" block is not available in the if block nor outside of the if/else control structure
  - If you need a variable to be defined in both blocks or *after* the 'if' statement, define it before the if/else statement
- You can nest if statements by using them inside other if / else statements (more on this next week)

Example Prog: Simple ‘if’ statement

```
package pearson;
import java.util.Scanner;

public class IfTest {

  public static void main (String[] args) {
    System.out.print("Please enter the value of x: ");

    Scanner input = new Scanner(System.in);   // Scanner to ask for input
    int x = input.nextInt();

    if (x == 3) {
      System.out.println("x must be 3");
    }

    System.out.println("This runs no matter what");
  }
}
```

Example Prog 2: ‘if’ statement with ‘else’

```
...
public class IfTest2 {

  public static void main (String[] args) {
    System.out.print("Please enter the value of x: ");

    Scanner input = new Scanner(System.in);   // Scanner to ask for input
    int x = input.nextInt();

    if (x == 3) {
       System.out.println("x is 3");
    }
    else {
       System.out.println("x is NOT 3");
    }
    System.out.println("This runs no matter what");
  }
}
```

Example Prog 3: ‘if’ statement with ‘else’ (Strings)

```
...
public class IfTest3 {

  public static void main (String[] args) {
    System.out.print("Please enter two letters of the alphabet: ");

    Scanner input = new Scanner(System.in);   // Scanner to ask for input
    String line= input.nextLine();

    if (line.equals("AB")) {
       System.out.println("String is AB");
    }
    else {
       System.out.println("String is NOT AB");
    }
    System.out.println("This runs no matter what");
  }
}
```