NYU

Courant Institute of Mathematical Sciences
Department of Computer Science
CS101 Introduction to Computer Science

Anasse Bari, Ph.D.

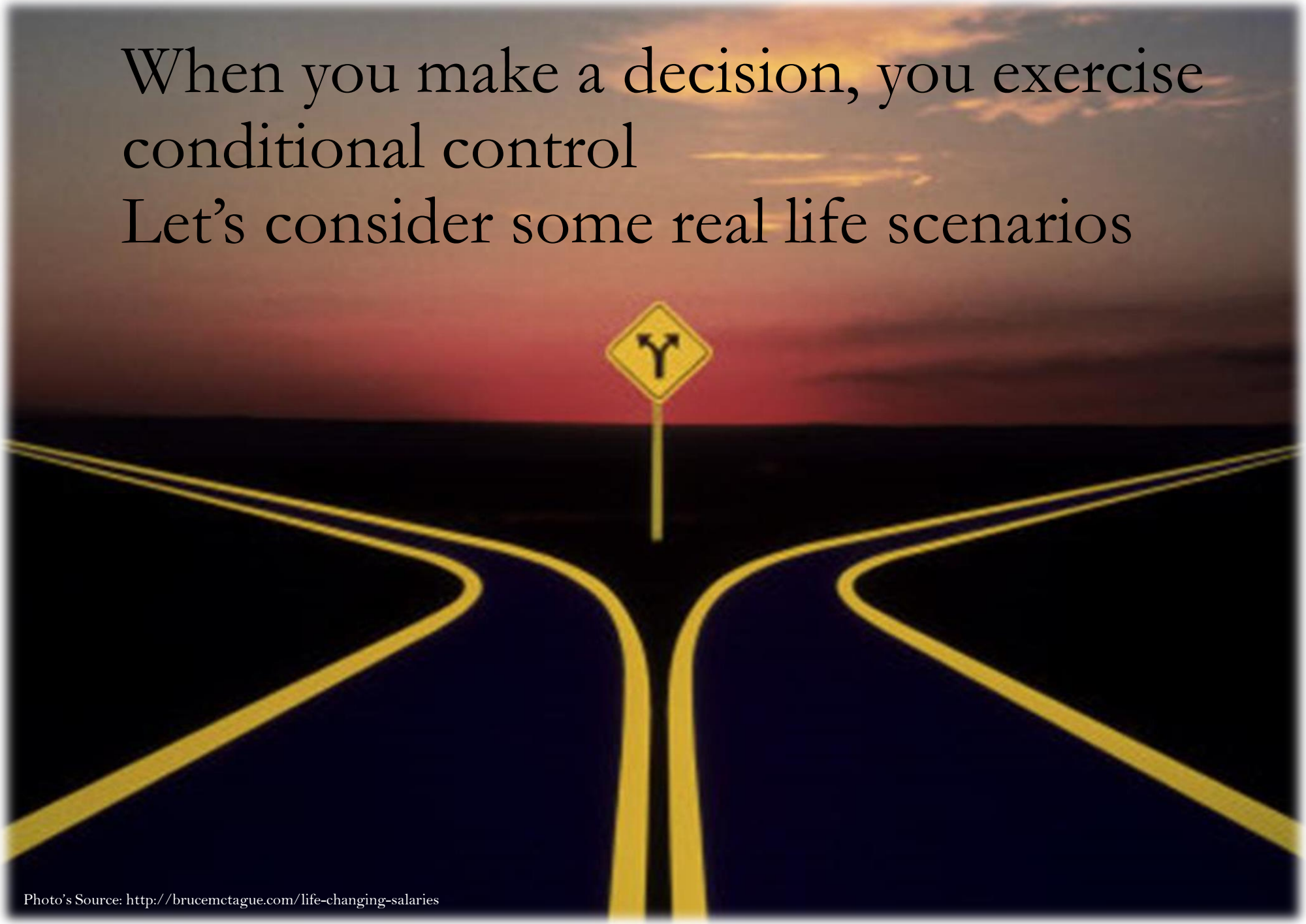
Chapter#5: Control Statements

Objectives

- Learning the basic three ways of processing statements in a programming language
- Introducing conditionals
- Learning if statement, if else statements…
- Learning the switch statement
- Introducing the how of writing conditions

When you make a decision, you exercise conditional control

Let's consider some real life scenarios



Photo's Source: http://brucemctague.com/life-changing-salaries

Three main ways of processing statements

- Three methods of processing statements in a program
  - In sequence
  - Branching (conditions)
  - Looping
- Branching: Altering the flow of program execution by making a selection or choice
- Looping: Altering the flow of program execution by repetition of a particular block of statement(s)

Decisions & Conditions

- Use a decision statement when an action is to be taken only if a particular condition holds
- The condition which must hold may be logical or relational expression or a Boolean variable. The value of the condition must be True or False
- Each possible path through a condition statement will contain a sequence of steps to be executed
- The condition and the sequences of steps that are executed for each outcome of the condition statement form a selection structure.
- A selection structure is a type of control structure

Recap: *Statement Types in Java*

- Programs in Java consist of a set of **classes**. Those classes contain **methods**, and each of those methods consists of a sequence of **statements**.

- Statements in Java fall into three basic types:
  - Simple statements
  - Compound statements
  - Control statements
- **Simple statements** are formed by adding a semicolon to the end of a Java expression.
- **Compound statements** (also called **blocks**) consist of a sequence of statements enclosed in curly braces.
- **Control statements** fall into two categories:
  - **Conditional statements** that specify some kind of test
  - **Iterative statements** that specify repetition

Reference: The Art and Science of Java: Eric Roberts:

- Control Statement
  - Problem: Buying a product at an Apple store
  - Condition: Enough budget ?

  Decisions:
    - If (I can afford it) I will buy it, otherwise I will not
    - Live Demo: BuyIphone.java (Next slide)

```java
import java.util.Scanner;
public class BuyAnIphone {

    public static void main (String [] args) {

        double budget;
        double price = 699.99;
        double change;
        double amountneeded;

        System.out.println ("What's your budget?");

        Scanner budgetScanner = new Scanner (System.in);

        budget = budgetScanner.nextDouble();

        if (budget >= price) {
        System.out.println("You can buy the iPhone");

        change = budget-price;
        System.out.println ("Your change: " +change);

        }else {
            System.out.println("You can't buy the iPhone");

            amountneeded = price-budget;
            System.out.println("You need: " +amountneeded);

        }

        if (budget == price) {
            System.out.println ("Your budget is equal to the price");
        }

}
```

The **if** Statement

The simplest of the control statements is the **if** statement, which occurs in two forms. You use the first form whenever you need to perform an operation only if a particular condition is true:

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

You use the second form whenever you want to choose between two alternative paths, one for cases in which a condition is true and a second for cases in which that condition is false:

```
if (condition) {
   statements to be executed if the condition is true
} else {
   statements to be executed if the condition is false
}
```

Reference: The Art and Science of Java: Eric Roberts:

Choosing between **if** and **if/else**

- As is true with most programming constructs, there is no hard-and-fast rule that will tell you whether you need the basic **`if`** statement or the **`if`**/**`else`** form.

- The best general guideline is to think about the English description of the problem you are trying to solve. If that description contains the words *else* or *otherwise*, there is a good chance that the solution will use the **`else`** keyword.

- Example: BuyIphone.java

Common Forms of the `if` Statement

The examples in the book use the **`if`** statement in the following forms:

Single line **`if`** statement

```
if (condition) statement
```

Multiline **`if`** statement with curly braces

```
if (condition) {
    statement
    . . . more statements . . .
}
```

`if`/**`else`** statement with curly braces

```
if (condition) {
    statements_true
} else {
    statements_false
}
```

Cascading **`if`** statement

```
if (condition_1) {
    statements_1
} else if (condition_2) {
    statements_2
. . . more else/if conditions . . .
} else {
    statements_else
}
```

Reference: The Art and Science of Java: Eric Roberts:

Example of Cascading **if** Statement

Notice several conditions!

Cascading `if` statement

```
if (condition1) {
    statements1
} else if (condition2) {
    statements2
. . . more else/if conditions . . .
} else {
    statementselse
}
```

Example of Cascading **if** Statement

Cascading **if** statement

if (*condition*$_1$) {
  statements$_1$
} else if (*condition*$_2$) {
  statements$_2$
. . . *more* **else**/**if** *conditions* . . .
} else {
  statements$_{\text{else}}$
}

Quiz Grades !
If the grade is between 18 and 20
If is an A
15 and 18 it is a B
Between 12-15 it is a C
Below 12
Come see me !

Cascading if statement example

```
import java.util.Scanner;
public class Quiz1 {

    public static void main(String[] args) {
        double score;
        Scanner scan = new Scanner(System.in);
        System.out.println("What was the given grade?");
        score = scan.nextDouble();
        if (score >= 0 && score <=20){
            if (score >=18){
                System.out.println("Congratulations! You have earned you
            }else if (score >= 15){
                    System.out.println("Good Job, you have earned a \"B\
            }else if (score>= 12){
                System.out.println("Not bad you have a \"C\"");
            }else {
                System.out.println("You need to see the Professor about
            }

        }else {
            System.out.println("Check your grade again, that does not co
        }


    }

}
```

Another way to do it …

```
import java.util.Scanner;
public class Quiz1 {

    public static void main(String[] args) {
        double score;
        Scanner scan = new Scanner(System.in);
        System.out.println("What was the given grade?");
        score = scan.nextDouble();
        if (score >= 0 && score <=20){
            if (score >=18){
                System.out.println("Congratulations! You have earned you
            }else if (score >= 15){
                    System.out.println("Good Job, you have earned a \"B\
            }else if (score>= 12){
                System.out.println("Not bad you have a \"C\"");
            }else {
                System.out.println("You need to see the Professor about
            }

        }else {
            System.out.println("Check your grade again, that does not co
        }


    }

}
```

The switch Statement

- The *switch statement* provides another way to decide which statement to execute next
- The switch statement evaluates an integral expression (mainly for int or char), then attempts to match the result to one of several possible *cases*
- *You can also use Switch for Strings* ***http://docs.oracle.com/javase/7/docs/technotes/guides/language/strings-switch.html***
- Each case contains a value and a statement list
- The flow of control transfers to the statement list associated with the first case value that matches

The switch Statement

- Often a *break statement* is used as the last statement in each case's statement list
- A break statement causes control to transfer to the end of the switch statement
- If a break statement is not used, the flow of control will continue into the next case
- Sometimes this may be appropriate, but often we only want to execute the statements associated with one case

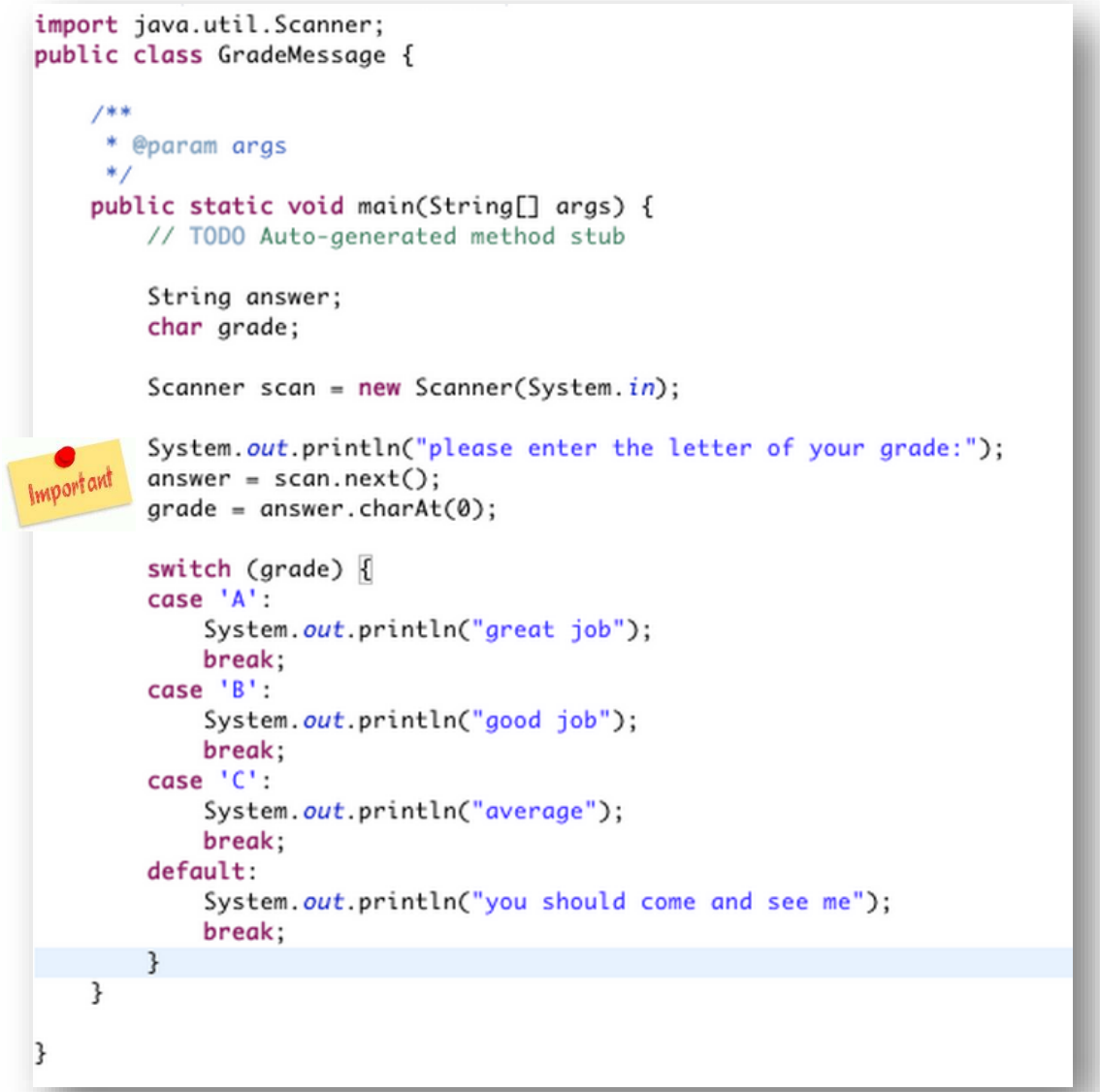
The **switch** Statement Syntax

The switch statement provides a convenient syntax for choosing among a set of possible paths:

```
switch ( expression ) {
  case v1:
    statements to be executed if expression = v1
    break;
  case v2:
    statements to be executed if expression = v2
    break;
  . . . more case clauses if needed . . .
  default:
    statements to be executed if no values match
    break;
}
```

Reference: The Art and Science of Java: Eric Roberts:

The switch Statement
Example




```java
import java.util.Scanner;
public class GradeMessage {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

        String answer;
        char grade;

        Scanner scan = new Scanner(System.in);

        System.out.println("please enter the letter of your grade:");
        answer = scan.next();
        grade = answer.charAt(0);

        switch (grade) {
        case 'A':
            System.out.println("great job");
            break;
        case 'B':
            System.out.println("good job");
            break;
        case 'C':
            System.out.println("average");
            break;
        default:
            System.out.println("you should come and see me");
            break;
        }
    }

}
```

The switch Statement

- A switch statement can have an optional *default case*
- The default case has no associated value and simply uses the reserved word default
- If there is a default case and no other value matches, control will transfer to the default statement list
- If there is no default case and no other value matches, control falls through to the statement after the switch without executing any statements

Important Example using Switch for Strings

Switch Statement for Strings
http://docs.oracle.com/javase/7/docs/technotes/guides/language/strings-switch.html

NYU

Courant Institute of Mathematical Sciences
Department of Computer Science
CS101 Introduction to Computer Science

Anasse Bari, Ph.D.

End of Chapter#5: Conditional Statements