ITERATION

(Chapter 3)

Iteration

- **Iteration** is the form of program control that allows us to repeat a section of code
- For this reason this form of control is often also referred to as **repetition**
- The programming structure that is used to control this repetition is often called a **loop**

- There are three types of loops in Java:
  - **for** loop
  - **while** loop
  - **do…while** loop

When to use a loop?

- Display a square of stars (5-by-5) on the screen

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

- This could be achieved with five output statements executed in sequence
- Better to **repeat** one output statement five times with a loop

“for” loop

- If we wish to repeat a section of code a *fixed number of times* (five in the example above) we would use Java's **for** loop
- The **for** loop is usually used in conjunction with a **counter** to keep track of how many times we have been through the loop so far

```
for( /* start counter */ ; /* test counter */ ; /* change counter */)
{
    // instruction(s) to be repeated go here
}
```

Display a 5-by-5 square of stars

```
for(int i = 1; i <= 5; i++)
{
   System.out.println("*****");
}
```

The loop counter

- Very common way of using a **for** loop:
  - start the counter at 1
  - add 1 to the counter each time the loop repeats
- However
  - you may start your counter at any value and
  - you may change the counter in anyway you choose

A different way of using the loop counter

```
public class Countdown
{
  public static void main(String[] args)
  {
    System.out.println("***Numbers from 10 to 1***");
    for (int i=10; i >= 1; i--)   // counter moving from 10 down to 1
    {
      System.out.println(i);
    }
  }
}
```

Nested loops

```
for(int i = 1; i <= 5; i++) // outer loop control
{
  for (int j = 1; j<=5; j++) // inner loop control
  {
    System.out.print("*");
  } // inner loop ends here
  System.out.println(); // next row on a new line
} // outer loop ends here
```

Non-fixed repetitions

- The **for** loop is an often used construct to implement fixed repetitions
- Sometimes a repetition is required that is *not fixed*
  - a racing game that repeatedly moves a car around a track until the car crashes
  - a ticket issuing program that repeatedly offers tickets for sale until the user chooses to quit the program
  - a password checking program that does not let a user into an application until he or she enters the right password
- The **while** loop offers one type of non-fixed iteration

“while” loop

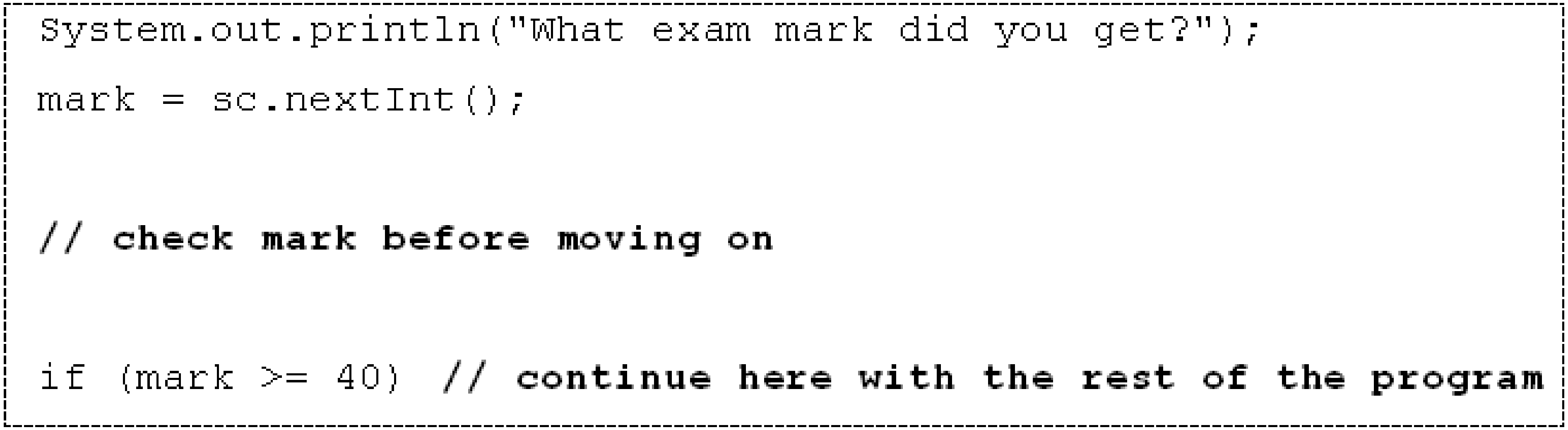
- The syntax for constructing this loop in Java is

```
while ( /* test goes here */ )
{
   // instruction(s) to be repeated go here
}
```

- As this loop is not repeating a fixed number of times, there is no need to create a counter to keep track of the number of repetitions

Input validation

- Checking input data for errors is referred to as **input validation**
  - For example, asking the user to enter an exam mark
    - It should never be greater than 100 or less than 0



```
System.out.println("What exam mark did you get?");
mark = sc.nextInt();

// check mark before moving on

if (mark >= 40) // continue here with the rest of the program
```

```
System.out.println("What exam mark did you get?");
mark = sc.nextInt();
while (mark < 0  || mark > 100) // check for invalid input
{
   // display error message and allow for re-input
   System.out.println("invalid mark: Re-enter!");
   mark = sc.nextInt();
}
if (mark >= 40)
// rest of code goes here
```

“do … while” loop

- The **do…while** loop is another variable loop construct
- Unlike the **while** loop, the **do…while** loop has its test at the *end* of the loop rather than at the *beginning*.
- The syntax of a **do…while** loop is given below:

```
do
{
   // instruction(s) to be repeated go here
}while ( /* test goes here */ ); // note semi-colon at the end
```

Implications of having the test at the end of the loop

- If the test is at the end of the loop, the loop will iterate *at least once*
- If the test is at the beginning of the loop, however, there is a possibility that the condition will be **false** to begin with, and the loop is never executed
- A **while** loop therefore executes *zero or more times*
- A **do...while** loop executes *one or more times.*

Using the “do … while” loop

- If you want to repeat the statements until the user chooses to quit your program, you can put the whole program in a loop
- Involves asking the user each time if he or she would like to continue repeating your program, or to stop
- A **while** loop would be difficult to use, as the test that checks the user's response to a question cannot be carried out at the beginning of the loop
- The answer is to move the test to the end of the loop and use a **do...while** loop

An example program

```
import java.util.*;

public class FindCost4
{
 public static void main(String[] args )
 {
  double price, tax;
  char reply;
  Scanner sc = new Scanner(System.in);
  do
  {
   // these instructions as before
   System.out.println("*** Product Price Check ***");
   System.out.print("Enter initial price: ");
   price = sc.nextDouble();
   System.out.print("Enter tax rate: ");
```

```
    tax = sc.nextDouble();
    price = price * (1 + tax/100);
    System.out.println("Cost after tax = " + price);


    // now see if user wants another go
    System.out.println();
    System.out.print
      ("Would you like to enter another product(y/n)?: ");
    reply = sc.next().charAt(0);
    System.out.println();
   } while (reply == 'y' || reply == 'Y');
  }
}
```

```
*** Product Price Check ***
Enter initial price: 50
Enter tax rate: 10
Cost after tax = 55.0
Would you like to enter another product (y/n)?: y


*** Product Price Check ***
Enter initial price: 70
Enter tax rate: 5
Cost after tax = 73.5
Would you like to enter another product (y/n)?: Y


*** Product Price Check ***
Enter initial price: 200
Enter tax rate: 15
Cost after tax = 230.0
Would you like to enter another product (y/n)?: n
```

Menu driven programs

- Another way to allow a program to be run repeatedly using a **do…while** loop is to include a *menu* of options within the loop
- The options themselves are processed by a **switch** statement
- One of the options in the menu list would be the option to quit and this option is checked in the **while** condition of the loop

Another example

```
import java.util.*;
public class TimetableWithLoop
{
  public static void main(String[] args)
  {
      char group, response;
      Scanner sc = new Scanner (System.in);
      System.out.println("***Lab Times***");
      do // put code in loop
      {
         // offer menu of options
```

```
System.out.println(); // create a blank line
System.out.println("[1] TIME FOR GROUP A");
System.out.println("[2] TIME FOR GROUP B");
System.out.println("[3] TIME FOR GROUP C");
System.out.println("[4] QUIT PROGRAM");
System.out.print("enter choice [1,2,3,4]: ");
response = sc.next().charAt(0); // get response
System.out.println(); // create a blank line
switch(response)  // process response
{
  case '1': System.out.println("10.00 a.m ");
            break;
  case '2': System.out.println("1.00 p.m ");
```

```
                    break;
          case '3': System.out.println("11.00 a.m ");
                    break;
          case '4': System.out.println("Goodbye ");
                    break;
          default:  System.out.println("Options 1-4 only!")
        }
      } while (response != '4'); // test for Quit option
    }
}
```

```
***Lab Times***
[1] TIME FOR GROUP A
[2] TIME FOR GROUP B
[3] TIME FOR GROUP C
[4] QUIT PROGRAM
enter choice [1,2,3,4]:  2

1.00 p.m

[1] TIME FOR GROUP A
[2] TIME FOR GROUP B
[3] TIME FOR GROUP C
[4] QUIT PROGRAM
enter choice [1,2,3,4]:  5

Options 1-4 only!

[1] TIME FOR GROUP A
[2] TIME FOR GROUP B
[3] TIME FOR GROUP C
[4] QUIT PROGRAM
enter choice [1,2,3,4]:  1

10.00 a.m
```

```
[1] TIME FOR GROUP A
[2] TIME FOR GROUP B
[3] TIME FOR GROUP C
[4] QUIT PROGRAM
enter choice [1,2,3,4]:  3

11.00 a.m

[1] TIME FOR GROUP A
[2] TIME FOR GROUP B
[3] TIME FOR GROUP C
[4] QUIT PROGRAM
enter choice [1,2,3,4]:  4
Goodbye
```

Picking the right loop

- Use a **for** loop
  - If the number of repetitions required can be determined prior to entering the loop -
- Use a **while** loop
  - If the number of repetitions required cannot be determined prior to entering the loop, and you wish to allow for the possibility of zero repetitions
- Use a **do...while** loop
  - If the number of repetitions required cannot be determined before the loop, and you require at least one repetition of the loop