Chapter 5 focuses on:
- conditional statements
- repetition (loop) statements

- boolean expressions
- comparing data

Today
- Additional loop and conditional statements
- More examples
Logic of a “for” loop

for ( initialization ; condition ; increment )
statement;

Stars.java (main method)

//------------------------------------------------------------------
//  Prints a ?? shape using asterisk (star) characters.
//------------------------------------------------------------------
public static void main (String[] args)
{
    final int MAX_ROWS = 10;

    for (int row = 1; row <= MAX_ROWS; row++)
    {
        for (int star = 1; star <= row; star++)
            System.out.print ("*");

        System.out.println();
    }
}
import java.util.Scanner;

public class Multiples {
    public static void main(String[] args) {
        final int PER_LINE = 5;
        int value, limit, mult, count = 0;
        Scanner scan = new Scanner(System.in);
        System.out.print("Enter a positive value: ");
        value = scan.nextInt();
        System.out.print("Enter an upper limit: ");
        limit = scan.nextInt();
        System.out.println();
        System.out.println("The multiples of " + value + " between " + value + " and " + limit + " (inclusive) are:");
        for (mult = value; mult <= limit; mult += value) {
            System.out.print(mult + "	");
            count++;
            if (count % PER_LINE == 0)
                System.out.println();
        }
    }
}

---

Logic of a while Loop

```
while (condition)
{
    statement;
}
```

- **condition evaluated**
- **true**
- **false**

- **statement**
System.out.println("Enter a potential palindrome:");
str = scan.nextLine();

left = 0;
right = str.length() - 1;

while (str.charAt(left) == str.charAt(right) && left < right)
{
    left++;
    right--;
}
System.out.println();

if (left < right)
    System.out.println("That string is NOT a palindrome.");
else
    System.out.println("That string IS a palindrome.");

The do Statement

- A do statement has the following syntax:
  do
  {
      statement;
  }
  while ( condition )

- The statement is executed once initially, and then the condition is evaluated

- The statement is executed repeatedly until the condition becomes false
The do Statement

- An example of a do loop:
  ```java
  int count = 0;
  do
  {
      count++;
      System.out.println (count);
  } while (count < 5);
  ```

- The body of a do loop executes at least once
- See [ReverseNumber.java](page 251): understand its algorithm
ReverseNumber (main method)

```java
// Reverses the digits of an integer mathematically.
public static void main (String[] args)
{
    int number, lastDigit, reverse = 0;
    Scanner scan = new Scanner (System.in);
    System.out.print("Enter a positive integer: ");
    number = scan.nextInt();
    do
    {
        lastDigit = number % 10;
        reverse = (reverse * 10) + lastDigit;
        number = number / 10;
    }
    while (number > 0);
    System.out.println("That number reversed is "+reverse);
}
```

Comparing while and do

<table>
<thead>
<tr>
<th>Condition Evaluated</th>
<th>Statement</th>
<th>Pre-Test Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Condition Evaluated</th>
<th>Statement</th>
<th>Post-Test Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>true</td>
<td>true</td>
<td></td>
</tr>
<tr>
<td>false</td>
<td>false</td>
<td></td>
</tr>
</tbody>
</table>
The Conditional Operator

- Java has a ***conditional operator*** that uses a boolean condition to determine which of two expressions is evaluated.

- Its syntax is:
  
  \[ \text{condition} \ ? \ \text{expression1} : \ \text{expression2} \]

- If the **condition** is true, **expression1** is evaluated; if it is false, **expression2** is evaluated.

- The value of the entire conditional operator is the value of the selected expression.
The Conditional Operator

- The conditional operator is similar to an if-else statement, except that it is an expression that returns a value.

- For example:

```
larger = ((num1 > num2) ? num1 : num2);
```

- If `num1` is greater than `num2`, then `num1` is assigned to `larger`; otherwise, `num2` is assigned to `larger`.

- The conditional operator is *ternary* because it requires three operands.

---

Another example:

```
System.out.println("Your change is ", count + ((count == 1) ? "Dime" : "Dimes");
```

- If `count` equals 1, then "Dime" is printed.
- If `count` is anything other than 1, then "Dimes" is printed.
The switch Statement

- An example of a switch statement:
  ```java
  switch (option) {
    case 'A':
      aCount++;
      break;
    case 'B':
      bCount++;
      break;
    case 'C':
      cCount++;
      break;
  }
  ```

- The `switch statement` provides another way to decide which statement to execute next.
- The `switch` statement evaluates an expression, then attempts to match the result to one of several possible `cases`.
- Each case contains a value and a list of statements.
- The flow of control transfers to statement associated with the first case value that matches.
The switch Statement

- The general syntax of a switch statement is:

```
switch (expression)
{
    case value1:
        statement-list1
    case value2:
        statement-list2
    case value3:
        statement-list3
    case ...
}
```

If expression matches value2, control jumps to here

- 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 want to execute only the statements associated with one case
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 the default case is present, control will transfer to it if no other case value matches
- If there is no default case, and no other value matches, control falls through to the statement after the switch

The switch Statement

- The expression of a `switch` statement must result in an `integral type`, meaning an integer (`byte, short, int, long`) or a `char`
- It cannot be a `boolean` value or a floating point value (`float` or `double`)
- The implicit boolean condition in a `switch` statement is equality
- You cannot perform relational checks with a `switch` statement
- See `GradeReport.java` (page 233)
```java
category = grade / 10;
System.out.print("That grade is ");
switch (category)
{
    case 10:
        System.out.println("a perfect score. Well done.");
        break;
    case 9:
        System.out.println("well above average. Excellent.");
        break;
    case 8:
        System.out.println("above average. Nice job.");
        break;
    case 7:
        System.out.println("average.");
        break;
    case 6:
        System.out.println("below average. You should see the");
        System.out.println("instructor to clarify the material " +"presented in class.");
        break;
    default:
        System.out.println("not passing.");
}
```