CS1007 lecture #6 notes

thu 19 sep 2002

- news
  - homework #2 will be posted by midnight tonight
  - homework #1 should be returned in recitation next week
- the `while` and `for` loops
- `break` and `continue` statements
- `switch` statement
- reading: *ch 3.5-3.10*

looping.

- if you want to do something many times
- two types of loops:
  - counter controlled (today)
  - condition controlled (next time)
- three loop statements:
  - `for`
  - `while`
  - `do`

counter-controlled `for` loop.

```
public class ex6a {
  public static void main ( String[] args ) {
    int n, i;
    n = (Integer.valueOf( args[0] )).intValue(); // String -> int
    System.out.println( "counting up to " + n + "..." );
    for ( i=0; i<n; i++ ) {
      System.out.print( i+ "  " );
    } // end for
    System.out.println();
  } // end of main
} // end of class ex6a
```

remember the increment and decrement operators...

- increment: ++

  ```
  i++;
  ```

  is the same as:

  ```
  i = i + 1;
  ```

- decrement: --

  ```
  i--;
  ```

  is the same as:

  ```
  i = i - 1;
  ```

counter-controlled `while` loop.

```
public class ex6b {
  public static void main ( String[] args ) {
    int n, i;
    n = (Integer.valueOf( args[0] )).intValue(); // String -> int
    System.out.println( "counting up to " + n + "..." );
    i = 0;
    while ( i<n ) {
      System.out.print( i+ "  " );
      i++;
    } // end while
    System.out.println();
  } // end of main
} // end of class ex6b
```

counter-controlled do loop.

```
public class ex6c {
  public static void main ( String[] args ) {
    int n, i;
    n = (Integer.valueOf( args[0] )).intValue(); // String -> int
    System.out.println( "counting up to " + n + "..." );
    i = 0;
    do {
      System.out.print( i+ "  " );
      i++;
    } while ( i<n );
    System.out.println();
  } // end of main
} // end of class ex6c
```

recall the `if` branching statement...

```
public class ex6d {
    public static void main ( String[] args ) {
        for ( int i=1; i<=10; i++ ) {
            System.out.print( i + ", " );
            if ( i == 2 ) {
                System.out.println( "buckle my shoe" );
            }
            else if ( i == 4 ) {
                System.out.println( "shut the door" );
            }
            else if ( i == 6 ) {
                System.out.println( "pick up sticks" );
            }
            else if ( i == 8 ) {
                System.out.println( "lay them straight" );
            }
            else if ( i == 10 ) {
                System.out.println( "a big fat hen" );
            } // end if-else
        } //end for i
    } // end main()
} // end of class ex6d
```

the `switch` branching statement .

```
public class ex6e {
  public static void main ( String[] args ) {
    for ( int i=1; i<=10; i++ ) {
      System.out.print( i + ", " );
      switch ( i ) {
      case 2:
        System.out.println( "buckle my shoe" );
        break;
      case 4:
        System.out.println( "shut the door" );
        break;
      case 6:
        System.out.println( "pick up sticks" );
        break;
      case 8:
        System.out.println( "lay them straight" );
        break;
      case 10:
        System.out.println( "a big fat hen" );
        break;
      } // end of switch
    } // end for i
  } // end main()
} // end of class ex6e
```

`if` with compound statements .

```
public class ex6e {
  public static void main ( String[] args ) {
    for ( int i=1; i<=10; i++ ) {
      System.out.print( i + ", " );
      switch ( i ) {
      case 2:
        System.out.println( "buckle my shoe" );
        break;
      case 4:
        System.out.println( "shut the door" );
        break;
      case 6:
        System.out.println( "pick up sticks" );
        break;
      case 8:
        System.out.println( "lay them straight" );
        break;
      case 10:
        System.out.println( "a big fat hen" );
        break;
      } // end of switch
    } // end for i
  } // end main()
} // end of class ex6e
```

compound `switch`.

```
public class ex6g {
    public static void main ( String[] args ) {
        for ( int i=1; i<=10; i++ ) {
            switch ( i ) {
            case 1:
            case 3:
            case 5:
            case 7:
            case 9:
                System.out.print( i + ", " );  break;
            case 2:
                System.out.println( i + ", buckle my shoe" );  break;
            case 4:
                System.out.println( i + ", shut the door" );  break;
            case 6:
                System.out.println( i + ", pick up sticks" );  break;
            case 8:
                System.out.println( i + ", lay them straight" );  break;
            case 10:
                System.out.println( i + ", a big fat hen" );  break;
            } // end of switch
        } // end for i
    } // end main()
} // end of class ex6g
```

`switch` with `default`.

```
public class ex6h {
  public static void main ( String[] args ) {
    for ( int i=1; i<=10; i++ ) {
      switch ( i ) {
        case 2:
          System.out.println( i + ", buckle my shoe" );
          break;
        case 4:
          System.out.println( i + ", shut the door" );
          break;
        case 6:
          System.out.println( i + ", pick up sticks" );
          break;
        case 8:
          System.out.println( i + ", lay them straight" );
          break;
        case 10:
          System.out.println( i + ", a big fat hen" );
          break;
        default:
          System.out.print( i + ", " );
        } // end of switch
      } // end for i
  } // end main()
} // end of class ex6h
```

break and continue.

- these statements interrupt the normal flow of control of a program
- `break` is used in the `switch` statement to jump out of a case clause, without dropping down into the next one
- `break` can also be used from within a loop to interrupt the loop and jump to the end of the loop
- `continue` is used from within a loop to interrupt the loop and jump to the next iteration of the loop
- in general, these statements are bad to use because they allow you to write code that jumps around and may be more prone to errors