Lesser-known Control Statements

- The **SWITCH** statement
- The **DO** statement
- The **infinite** loop
- The iterative **FOR** loop

4.4 – The **switch** Statement

```java
System.out.print("Enter a numeric grade (0 to 100): ");
grade = scan.nextInt();
category = grade / 10;
System.out.print("That grade is ");
switch (category)
{
case 10:
    System.out.println("A. Excellent.");
    break;
case 9:
    System.out.println("B. Very Good.");
    break;
case 8:
    System.out.println("C. Good.");
    break;
case 7:
    System.out.println("D. Needs work.");
    break;
default:
    System.out.println("F. Not a passing grade.");
}
```

- The expression of a switch statement MUST result in an integral type, i.e., an integer (byte, short, int, long) or a char
- It CANNOT be a boolean or a floating point value (float or double)

4.5 – Infinite Loops

- An example of an infinite loop

```java
int count = 1;
while (count <= 25) {
    System.out.println (count);
    count = count - 1;
}
```

- This loop will continue executing until interrupted (Control-C) or until an underflow error occurs
- But why would you ever use an infinite loop?
4.7 – Comparing while and do

The while Loop
- Condition evaluated
- True
- Statement
- False

The do Loop
- Condition evaluated
- True
- Statement
- False

```java
int count = 0;
do {
count++;
process (count);
} while(count < 5);
```

4.6 – Iterators

- An iterator is an object that allows you to process a collection of items one at a time.
  - An iterator object has a hasNext method that returns true if there is at least one more item to process.
  - The next method returns the next item.
- The scanner class is an iterator
  - The hasNext method returns true if there is more data to be scanned.
  - The next method returns the next scanned token as a string.
- The scanner class also has variations on the hasNext method for specific data types (such as hasNextInt).

4.8 – Iterators and for Loops

- A variant of the for loop simplifies the repetitive processing of items.
- For example, if BookList is an iterator that manages Book objects, the following loop will print each book:

  ```java
  for (Book myBook : BookList)
  System.out.println (myBook);
  ```

- This style of for loop can be read “for each Book in BookList, ...”
- Therefore the iterator version of the for loop is sometimes referred to as the for each loop.
- No need to call the hasNext and next methods explicitly.

Splitting your input with a delimiter

```java
fileScan = new Scanner (new File("websites.inp"));
new// Read and process each line of the file
while (fileScan.hasNext())
{
url = fileScan.nextLine();
System.out.println ("URL: " + url);
urlScan = new Scanner (url);
urlScan.useDelimiter ("/");
// Print each part of the url
while (urlScan.hasNext())
{
System.out.println (" + urlScan.next());
}
System.out.println();
}
```