10

Control Flow – Branching
• This control has three statements, as follow:
  1. Break
  2. Continue
  3. Return
break statement

- As the name says, Break Statement is generally used to break the loop or switch statement.
- The break statement has two forms: labeled and unlabeled.
- You can also use an unlabeled break to terminate a for, while, or do-while loop, as shown in the following BreakDemo program:
Branch statement

- the **branching statements** (break, continue, return) supported by the Java programming language.
- Branching/Transfer/Jump
Unlabeled break statement

- The unlabeled version of the break statement is used when we want to jump out of a single loop or single case in switch statement.

- You saw the unlabeled form in the previous discussion of the switch statement.

```java
switch (dayCode) {
    case 0: dayStr = "Sunday";
        break;
    case 1: dayStr = "Monday";
        break;
    case 2: dayStr = "Tuesday";
        break;
    case 3: dayStr = "Wednesday";
        break;
    case 4: dayStr = "Thursday";
        break;
    case 5: dayStr = "Friday";
        break;
    case 6: dayStr = "Saturday";
        break;
    default: dayStr = "Invalid day code!";
        break;
}
```
Labeled break statement

- Labeled version of the break statement is used when we want to jump out of nested or multiple loops.
public class BreakLabeledEnum {

    enum Week {
        SUNDAY, MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY, SATURDAY
    }

    public static void main(String args[]) {
        int searchForWeekOrdinal = 0;
        int i = 0;
        boolean foundIt = false;
        String foundDay = "";

        System.out.println("Here are all week constants" + " and their ordinal values: ");
        System.out.println("-----");
        for (Week day : Week.values())
            System.out.println(day.ordinal() + " : " + day);
        System.out.println("-----");
    }
}
Class BreakLabeledEnum 2/3

```java
search:
    for (Week day : Week.values()){
        if (day.ordinal() == searchForWeekOrdinal) {
            foundIt = true;
            foundDay = day.toString();
            break search;
        } // if
    } // for
// search

if (foundIt){
    System.out.println("Found! " + searchForWeekOrdinal + " in the range = " + foundDay );
}
else {
    System.out.println("Not found! " + searchForWeekOrdinal + " not in the range!" );
}
} // main
} // class
```
Here are all week constants and their ordinal values:

0  :  SUNDAY
1  :  MONDAY
2  :  TUESDAY
3  :  WEDNESDAY
4  :  THURSDAY
5  :  FRIDAY
6  :  SATURDAY

Found! 0 in the range! = SUNDAY
This program searches for the ordinal = 0 in a range of days. The break statement, shown in boldface, terminates the for loop when that value is found. Control flow then transfers to the print statement at the end of the program. This program's output is:

- Found! 0 in the range! = SUNDAY
continue statement

- Continue statement is used when we want to skip the rest of the statement in the body of the loop and continue with the next iteration of the loop.
- The `continue` keyword can be used in any of the loop control structures. It causes the loop to immediately jump to the next iteration of the loop.
- In a for loop, the `continue` keyword causes flow of control to immediately jump to the update statement.
- In a while loop or do/while loop, flow of control immediately jumps to the Boolean expression.
- There are two forms of `continue` statement in Java.
  1. Unlabeled Continue Statement
  2. Labeled Continue Statement
length() : Returns the length of the sequence of characters (given string).

TIP - String class is zero-indexed, the range 0..n-1
Unlabeled continue statement

The string = peter piper picked a peck of pickled peppers
Found 9 p's in the string.
Labeled continue statement

```java
package Package05;

public class ContinueLabel {

    public static void main(String[] args) {

        outer: // label
        for (int i=0; i<10; i++) {
            for(int j=0; j<10; j++) {
                if(j > i) {
                    System.out.println();
                    continue outer;
                }

                System.out.print(" "+(i * j));
            }
        }

        System.out.println();
    }
}
```
Here is an example program that uses `continue` to print a triangular multiplication table for 0 through 9.
return statement

- The `return` statement is used to explicitly return from a method. That is, it causes program control to transfer back to the caller of the method. As such, it is categorized as a jump statement.
Return

```java
package Package05;

public class ReturnDemo {

    public static String NAME = "Java Sanjaya",
                          CITY = "Palembang";

    public static String getName() {
        return NAME;
    } // getName

    public static String getCity() {
        return CITY;
    } // getCity

    public static void main(String[] args) {
        String theName = getName(),
                        theCity = getCity();
        System.out.println(theName + " was born in " + theCity);
    } // main

} // class
```
<terminated> ReturnDemo [Java Application] C:
Java Sanjaya was born in Palembang