CSE310: Programming in Java

Topic: Branching Statements

Outlines

- break Statement
- continue Statement
- return Statement

break Statement

- break statement:
  - terminates a statement sequence in a switch statement
  - used to exit a loop

- The break statement has two forms:
  - labeled
  - unlabeled.

Unlabeled break

- An unlabeled break is used to terminate a for, while, or do-while loop and switch statement.

Example 1:

```
class Example
{
public static void main(String[] args)
{
        for(int i=0; i<100; i++)
                {
                if(i == 10)
                break;
        System.out.println("i: " + i);
                }
        System.out.println("Loop completed");
}
}
```

Labeled break Statement

- Java defines an expanded form of the break statement.

  break *label;*
- By using this form of break, we can break out of one or more blocks of code.
- When this form of break executes, control is transferred out of the named block.
- When this **break statement** is encountered with the *label/name of the loop*, it *skips* the execution any statement after it and takes the control right out of this labelled loop.

  And, the control goes to the first statement right after the loop.

Concept

```
outer:  while(condition)
        {
            if(condition)
              break outer;

            statement2;
        }
statement3;
```

while loop is labelled as "outer" and hence this statement "break outer" breaks the control out of the loop named "outer", without executing statement2.

labelled break

```
class Example
{
public static void main(String[] args)
        {
        outer:
        for(int i=0; i<3; i++)
        {
        System.out.println("Outer "+ i);
        inner:
        for(int j=0; j<3; j++)
        {
        System.out.println("Inner "+j);
        if(i== j+1)
        break outer;
        System.out.println("Bye");
        }
        }
        }
}
```

Output:

```
Outer 0
Inner 0
Bye
Inner 1
Bye
Inner 2
Bye
Outer 1
Inner 0
```

NOTE

- The break statement terminates the labeled statement; it does not transfer the flow of control to the label.

- Control flow is transferred to the statement immediately following the labeled (terminated) statement.

continue Statement

- The continue statement skips the current iteration of a for, while , or do-while loop.

- The unlabeled form skips to the end of the innermost loop's body and evaluates the boolean expression that controls the loop.

```
class Example
{
public static void main(String[] args)
  {
            int i;
            for(i=1;i<=10;i++)
            {
             if(i==5)
             continue;
            System.out.println(i);
            }
    }
}
```

Output:

1

2

3

4

6

7

8

9

10

Labeled continue Statement

- A labeled continue statement skips the current iteration of an outer loop marked with the given label.
- In **Labelled Continue Statement**, we give a *label/name* to a loop

When this continue statement is encountered with the label/name of the loop, it skips the execution any statement within the loop for the current iteration and *continues* with the next iteration and condition checking in the *labelled loop*.

Concept

```
outer:  for(initialization; conditon; iteration)
        {
            for(initialization; conditon; iteration)
            {
              if(condition)
                 continue outer;
            }
            statement2;
        }
statement3;
```

Outer for loop is labelled as "**outer**" and hence this statement "**continue outer**" bypasses the execution of statement2 & continues with the loop named "**outer**" & it's next iteration &condition check

labelled continue

Example

```
public class Main
{
  public static void main (String[]args)
  {
  loop:
    for (int i = 0; i < 2; i++)
      {
                for (int j = 0; j < 5; j++)
                  {
                    if (j == 2)
                      continue loop;
                      System.out.println ("i =" + i + " j =" + j);
                  }
      }
    System.out.println ("Out of the loop");
  }
}
```

Output:

i=0 j=0

i=0 j=1

i=1 j=0

i=1 j=1

Out of the loop

return Statement

- The return statement exits from the current method, and control flow returns to where the method was invoked.

- The return statement has two forms: one that returns a value, and one that doesn't.

- To return a value, simply put the value (or an expression that calculates the value) after the return keyword.

```
return ++count;
```

return Statement

- The data type of the returned value must match the type of the method's declared return value.

- When a method is declared void, use the form of return that doesn't return a value.

```
return;
```

Output??

```
public class Main
{
  public static void main (String[]args)
  {
  label:
for(int i=0; i<2; i++)
 {
for(int j=0; j<2; j++)
{
System.out.println(i + ", "+ j);
  if(j!=2)
 continue label;
}
 }
  }
}
```

A. 0,0
   1,0
B. 1,1
C. 0,0
D. 0,1
   1,1

Ans: A

Output??

```
public class Main
{
  public static void main (String[]args)
  {
  label:
    for (int i = 0; i < 2; i++)
      {
                for (int j = 0; j < 2; j++)
                  {
                    if (j>0)
                      break label;
                    System.out.print(i+" ");
                  }
      }
  }
}
```

A. 1

B. 0 1

C. 1 0

D. 0

Ans:D

Output??

```
public class Main
{
  public static void main (String[]args)
  {
  label1:
    for (int i = 1; i <=2; i++)
      {
  label2:
              for (int j = 2; j <=5; j++)
                {
                  System.out.print(j+" ");
                  if (j%2==0)
                    break label1;
                    else
                    continue label2;
                }

      }
  }
}
```

A. 2

B. 1 3 5

C. 1 3

D. 3 5

Ans:A

Output??

```
public class Main
{
  public static void main (String[]args)
  {
     int i=1,j=1;
  label1:
   while(i<=2)
   {
      i++;
                while(j<=2)
                {
                j++;
                System.out.println(i);
                if(i==j)
                break label1;
                }
     }
  }
}
```

A. 1

B. 2

C. 1 2

D. 2 2

Ans:B