Chapter 3 Control Statements

1. No output. Note: else matches the most recent if.

2. No output.

3. a, c, and d are the same.

4.
```
if (number % 2 == 0)
  System.out.println("The number is even");
else
  System.out.println("The number is odd");
```

5. Switch variables must be of `char`, `byte`, `short`, or `int` data types. If a `break` statement is not used, the next `case` statement is performed. You can always convert a `switch` statement to an equivalent `if` statement, but not an `if` statement to a `switch` statement. The use of the `switch` statement can improve readability of the program in some cases. The compiled code for the `switch` statement is also more efficient than its corresponding `if` statement.

6. y is 2.

7.
```
switch (a) {
  case 1: x += 5; break;
  case 2: x += 10; break;
  case 3: x += 16; break;
  case 4: x += 34;
}
```

8. y is -1.

9. The loop body is executed nine times. The printout is `2, 4, 6, 8` on separate lines.

10. The difference between a `do-while` loop and a `while` loop is the order of evaluating the `continuation-condition` and executing the loop body. In a `while` loop, the `continuation-condition` is checked and then, if true, the loop body is executed. In a `do-while` loop, the loop body is executed for the first time before the `continuation-condition` is evaluated.

11. Same. When the i++ and ++i are used in isolation, their effects are same.

12. The three parts in a `for` loop control are as follows:

The first part initializes the control variable.
The second part is a Boolean expression that determines whether the loop will repeat.
The third part is the adjustment statement, which adjusts the control variable.

```
for (int i=1,i<=100,i++)
  System.out.println(i);
```

13. The loop keeps doing something indefinitely.

14 No. The scope of the variable is inside the loop.

15. Yes. The advantages of `for` loops are simplicity and readability. Compilers can produce more efficient code for the `for` loop than for the corresponding `while` loop.

16. `while` loop:

```
long sum = 0;
int i=0;
while (i<=1000) {
  sum += i++;
}
```

`do-while` loop:

```
long sum = 0;
int i = 0;
do {
  sum += i++;
}
while (i <= 1000);
```

17. The keyword `break` is used to exit the current loop. The program in this example will terminate. The output is *Balance is 1.*

18. The keyword `continue` causes the rest of the loop body to be skipped for the current iteration. This `while` loop will not terminate.

19. Yes.

```
for (int i=1; sum<10000; i++)
   sum = sum + i;
```

20.

```
class TestBreak {
  public static void main(string[]args) {
    int sum = 0;
    int item = 0;

    do {
        item++;
        sum += item;
      }
      while(item<5 || sum >=6)
      System.out.println("The sum is "+sum);
    }
  }

  class TestContinue {
    public static void main(String[] args) {
      int sum = 0;
      int item = 0;

      do {
        item++;
        if (!(item == 2))
          sum += item;
      } while (item < 5);

      System.out.println("The sum is " + sum);
    }
  }
```

21. c.

22. a.

23. The semicolon (;) at the end of the for loop heading should be removed, and the semicolon (;) at the end of the if statement should be removed. Missing a semicolon for the first println statement. The semicolon (;) at the end of the while heading should be removed. Missing a semicolon at the end of the while loop.

24. i is not initialized.

25. The ; at the end of for loop should be removed.

```
 for (int i = 0; i < 10; i++);
```