Chapter 4 Loops

1. (A) The loop body is not executed.
   (B) The loop body is executed nine times. The printout is `2, 4, 6, 8` on separate lines.

2. 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.

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

4. 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);
```

5. The loop keeps doing something indefinitely.

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

7. 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.

8. `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);
```

9. No. Try n1 = 3 and n2 =3.

10. The keyword `break` is used to exit the current loop. The program in (A) will terminate. The output is *Balance is 1*.

The keyword `continue` causes the rest of the loop body to be skipped for the current iteration. The `while` loop will not terminate in (B).

11. Yes.

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

12. If a continue statement is executed inside a for loop, the rest of the iteration is skipped, then the action-after-each-iteration is performed and the loop-continuation-condition is checked. If a continue statement is executed inside a while loop, the rest of the iteration is skipped, then the loop-continuation-condition is checked.

Here is the fix:

```
int i = 0;

while (i < 4) {
  if (i % 3 == 0) {
    i++;
    continue;
  }
  sum += i;
  i++;
}
```

13.
```
class TestBreak {
  public static void main(string[]args) {
    int sum = 0;
    int number = 0;

    do {
      number++;
      sum += number;
    }
    while(number < 20 || sum >= 100);
    System.out.println("The sum is " + sum);
  }
}

class TestContinue {
  public static void main(String[] args) {
```

```
        int sum = 0;
        int number = 0;

        do {
          number++;
          if (number != 10 && number != 11)
            sum += number;
        } while (number < 20);

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

14. The statement labeled next.

15. The control is in the outer loop, and the next iteration of the outer loop is executed.

16. Line 3: The semicolon (;) at the end of the for loop heading should be removed.
Line 4: sum not defined.
Line 5: the semicolon (;) at the end of the if statement should be removed.
Line 6: Missing a semicolon for the first println statement.
Line 6: j not defined.
Line 10: The semicolon (;) at the end of the while heading should be removed.
Line 17: Missing a semicolon at the end of the while loop.

17. (A) compile error: i is not initialized.

(B) Line 3: The ; at the end of for loop should be removed.

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

18.

(A).

0 0 1 0 1 2 0 1 2 3

(B).

```
****
****
2 ****
3 2 ****
4 3 2 ****
```

(C).

1xxx2xxx4xxx8xxx16xxx

1xxx2xxx4xxx8xxx
1xxx2xxx4xxx
1xxx2xxx
1xxx

(D).

1G
1G3G
1G3G5G
1G3G5G7G
1G3G5G7G9G

19.

(A)

```
public class Test {
  public static void main(String[] args) {
    int i = 0;
    if (i > 0)
      i++;
    else
      i--;

    char grade;

    if (i >= 90)
      grade = 'A';
    else if (i >= 80)
      grade = 'B';
  }
}
```

(B)

```
public class Test {
  public static void main(String[] args) {
    for (int i = 0; i < 10; i++)
      if (i > 0)
        i++;
      else
        i--;
  }
}
```