Chapter 4 Programming Exercises - Solutions

# Exercise 4.1 - Find the Errors

a)

Error: Semicolon after while condition causes an empty loop; unmatched closing brace.

Correction:  
int i = 1;  
while (i <= 10) {  
 ++i;  
}

b)

Error: Comparing floating-point values with != can cause precision issues.

Correction:  
for (int i = 1; i <= 10; i++) {  
 double k = i \* 0.1;  
 System.out.println(k);  
}

c)

Error: Missing break statement after case 1 causes fall-through.

Correction:  
switch (n) {  
 case 1:  
 System.out.println("The number is 1");  
 break;  
 case 2:  
 System.out.println("The number is 2");  
 break;  
 default:  
 System.out.println("The number is not 1 or 2");  
 break;  
}

d)

Error: Condition should be <= to include 10.

Correction:  
int n = 1;  
while (n <= 10)  
 System.out.println(n++);

# Exercises 4.5 to 4.10 - Theory and Corrections

4.5 Four Basic Elements of Counter-Controlled Repetition:  
1. Initialization  
2. Condition  
3. Update  
4. Action

4.6 Compare `while` and `for`:  
- while: used when number of repetitions is unknown.  
- for: used when number of repetitions is known.

4.7 Use `do...while` when loop must run at least once before condition is tested (e.g., menu input validation).

4.8 `break` vs `continue`:  
- break: exits loop immediately.  
- continue: skips to next iteration.

4.9 a)  
Correction:  
for (int i = 100; i >= 1; i--)  
 System.out.println(i);

b)  
Correction:  
switch (value % 2) {  
 case 0:  
 System.out.println("Even integer");  
 break;  
 case 1:  
 System.out.println("Odd integer");  
 break;  
}

c)  
Correction:  
for (int i = 19; i >= 1; i -= 2)  
 System.out.println(i);

d)  
Correction:  
int counter = 2;  
do {  
 System.out.println(counter);  
 counter += 2;  
} while (counter <= 100);

4.10 Program prints a 10x5 block of '@':  
@@@@@  
@@@@@  
...  
@@@@@

# Exercises 4.11 to 4.14 - Applications

4.11 Find the Smallest Value:  
import java.util.Scanner;  
  
public class SmallestValue {  
 public static void main(String[] args) {  
 Scanner input = new Scanner(System.in);  
 System.out.print("Enter number of integers: ");  
 int count = input.nextInt();  
 int smallest = Integer.MAX\_VALUE;  
  
 for (int i = 0; i < count; i++) {  
 System.out.print("Enter number: ");  
 int num = input.nextInt();  
 if (num < smallest)  
 smallest = num;  
 }  
  
 System.out.println("Smallest number is: " + smallest);  
 }  
}

4.12 Product of Odd Integers from 1 to 15:  
public class ProductOfOdds {  
 public static void main(String[] args) {  
 long product = 1;  
 for (int i = 1; i <= 15; i += 2)  
 product \*= i;  
  
 System.out.println("Product of odd integers from 1 to 15: " + product);  
 }  
}

4.13 Factorials from 1 to 20:  
public class Factorials {  
 public static void main(String[] args) {  
 System.out.printf("%-5s %s%n", "n", "n!");  
 long factorial = 1;  
  
 for (int i = 1; i <= 20; i++) {  
 factorial \*= i;  
 System.out.printf("%-5d %d%n", i, factorial);  
 }  
 }  
}  
  
Note: Calculating 100! would exceed the range of type long. You would need to use BigInteger for larger values.

4.14 Modified Compound-Interest Program:  
public class Interest {  
 public static void main(String[] args) {  
 double principal = 1000.0;  
 int years = 10;  
  
 for (int rate = 5; rate <= 10; rate++) {  
 System.out.printf("Interest Rate: %d%%%n", rate);  
 for (int year = 1; year <= years; year++) {  
 double amount = principal \* Math.pow(1 + rate / 100.0, year);  
 System.out.printf("Year %2d: %.2f%n", year, amount);  
 }  
 System.out.println();  
 }  
 }  
}