***Patuakhali Science and Technology University***

Assignment on

# “Deitel Book Chapter - 6 and Section 4.2 solve”

Course Code: CCE-121

Course Title: Object Oriented Programming

Level - I; Semester - II

**<u>Submitted By</u>**

**Name: M.D. Sakibul Islam Shovon**

**ID:** 2302056

**REG:** 11834

**Session:** 2023-2024

Faculty of Computer Science and Engineering

**<u>Submitted To</u>**

**Prof. Dr. Md. Samsuzzaman**

Professor of Computer and Communication Engineering Department

Faculty of Computer Science and Engineering

# Deitel Book Chapter – 6 problem solve(6.1 – 6.6)

## 6.1 Fill in the blanks in each of the following statements:

**a)** A method is invoked with a(n) **method call**.

**b)** A variable known only within the method in which it's declared is called a(n) **local variable**.

**c)** The **return** statement in a called method can be used to pass the value of an expression back to the calling method.

**d)** The keyword **void** indicates that a method does not return a value.

**e)** Data can be added or removed only from the **top** of a stack.

**f)** Stacks are known as **LIFO (Last-In, First-Out)** data structures; the last item pushed (inserted) onto the stack is the first item popped (removed) from the stack.

**g)** The three ways to return control from a called method to a caller are **return**, **throw an exception**, and **method termination (end of method)**.

**h)** An object of class **SecureRandom** produces truly random numbers.

**i)** The method-call stack contains the memory for local variables on each invocation of a method during a program's execution. This data, stored as a portion of the method-call stack, is known as the **activation record** or **stack frame** of the method call.

**j)** If there are more method calls than can be stored on the method-call stack, an error known as a(n) **stack overflow** occurs.

**k)** The **scope** of a declaration is the portion of a program that can refer to the entity in the declaration by name.

**l)** It's possible to have several methods with the same name that each operate on different types or numbers of arguments. This feature is called method **overloading**.

## 6.2 For the class Craps in Fig. 6.8, state the scope of each of the following entities:

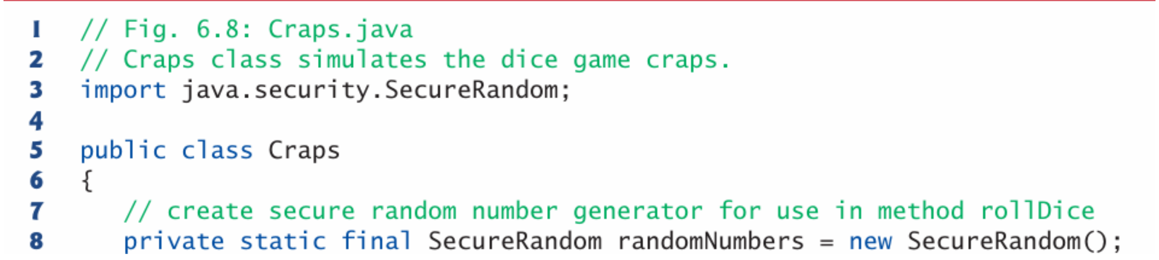
### a) the variable randomNumbers.

**b) the variable die1.**

**c) the method rollDice.**

**d) the method main.**

**e) the variable sumOfDice.**



```java
// Fig. 6.8: Craps.java
// Craps class simulates the dice game craps.
import java.security.SecureRandom;

public class Craps
{
   // create secure random number generator for use in method rollDice
   private static final SecureRandom randomNumbers = new SecureRandom();

   // enum type with constants that represent the game status
   private enum Status { CONTINUE, WON, LOST };

   // constants that represent common rolls of the dice
   private static final int SNAKE_EYES = 2;
   private static final int TREY = 3;
   private static final int SEVEN = 7;
   private static final int YO_LEVEN = 11;
   private static final int BOX_CARS = 12;

   // plays one game of craps
   public static void main(String[] args)
   {
      int myPoint = 0; // point if no win or loss on first roll
      Status gameStatus; // can contain CONTINUE, WON or LOST

      int sumOfDice = rollDice(); // first roll of the dice

      // determine game status and point based on first roll
      switch (sumOfDice)
      {
         case SEVEN: // win with 7 on first roll
         case YO_LEVEN: // win with 11 on first roll
            gameStatus = Status.WON;
            break;
         case SNAKE_EYES: // lose with 2 on first roll
         case TREY: // lose with 3 on first roll
         case BOX_CARS: // lose with 12 on first roll
            gameStatus = Status.LOST;
            break;
         default: // did not win or lose, so remember point
            gameStatus = Status.CONTINUE; // game is not over
            myPoint = sumOfDice; // remember the point
            System.out.printf("Point is %d%n", myPoint);
            break;
      }

      // while game is not complete
      while (gameStatus == Status.CONTINUE) // not WON or LOST
      {
         sumOfDice = rollDice(); // roll dice again

         // determine game status
         if (sumOfDice == myPoint) // win by making point
            gameStatus = Status.WON;
         else
            if (sumOfDice == SEVEN) // lose by rolling 7 before point
               gameStatus = Status.LOST;
      }

      // display won or lost message
      if (gameStatus == Status.WON)
         System.out.println("Player wins");
      else
         System.out.println("Player loses");
   }

   // roll dice, calculate sum and display results
   public static int rollDice()
   {
      // pick random die values
      int die1 = 1 + randomNumbers.nextInt(6); // first die roll
      int die2 = 1 + randomNumbers.nextInt(6); // second die roll

      int sum = die1 + die2; // sum of die values

      // display results of this roll
      System.out.printf("Player rolled %d + %d = %d%n",
         die1, die2, sum);

      return sum;
   }
} // end class Craps
```

```
Player rolled 5 + 6 = 11
Player wins
```

```
Player rolled 5 + 4 = 9
Point is 9
Player rolled 4 + 2 = 6
Player rolled 3 + 6 = 9
Player wins
```

```
Player rolled 1 + 2 = 3
Player loses
```

```
Player rolled 2 + 6 = 8
Point is 8
Player rolled 5 + 1 = 6
Player rolled 2 + 1 = 3
Player rolled 1 + 6 = 7
Player loses
```

## Solution:

a) the variable randomNumbers

- Declared as private static final SecureRandom randomNumbers = new SecureRandom(); inside the class.

- Scope: Entire class Craps. Since it’s private static final, it can be accessed anywhere inside Craps, but not outside the class.

b) the variable die1

- Declared inside method rollDice:

```
int die1 = 1 + randomNumbers.nextInt(6);
```

- Scope: Only inside rollDice method body, from its declaration line until the method ends.

c) the method rollDice

- Declared as a static method inside Craps.
- Scope: Can be called from any method within class Craps (like main), and also from outside the class if referred to as Craps.rollDice() (since it’s not private).

d) the method main

- Declared as public static void main(String[] args).
- Scope: The JVM calls it as the program entry point. Like rollDice, it belongs to the whole class, but it’s special since execution starts from here.

e) the variable sumOfDice

- Declared in main:

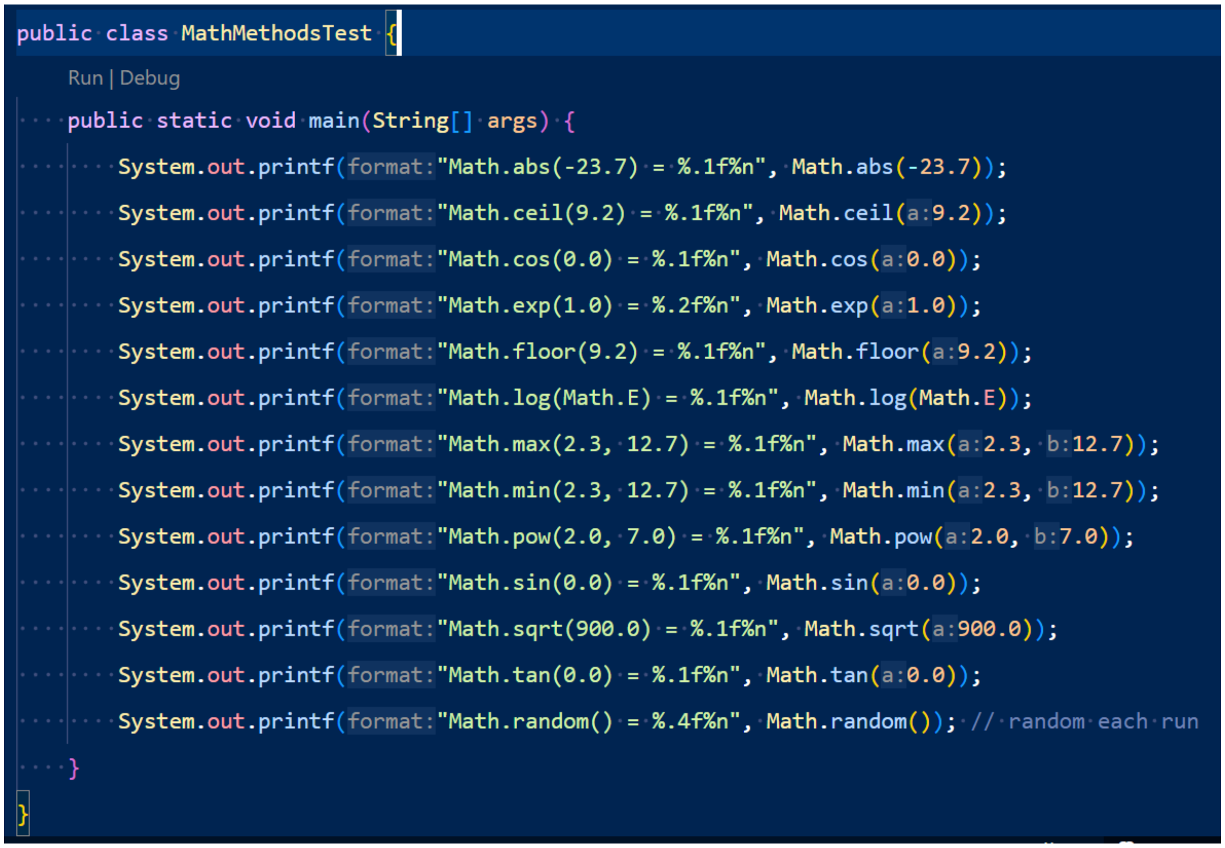
```
int sumOfDice = rollDice();
```

- Scope: Only inside main method, from declaration until the method ends.

## 6.3 <u>Write an application that tests whether the examples of the Math class method calls shown in Fig. 6.2 actually produce the indicated results.</u>

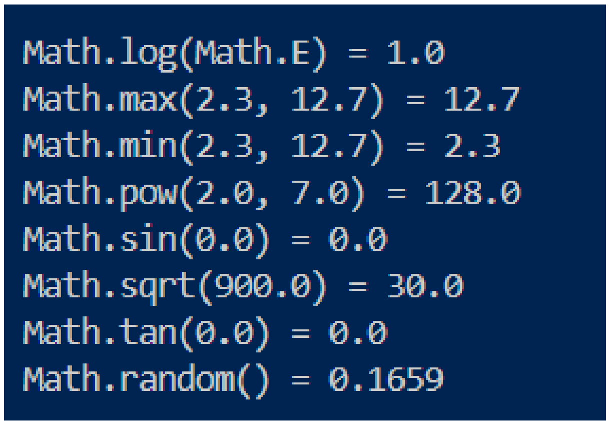
### Solution:

Code:



```java
public class MathMethodsTest {
    Run | Debug
    public static void main(String[] args) {
        System.out.printf(format:"Math.abs(-23.7) = %.1f%n", Math.abs(-23.7));
        System.out.printf(format:"Math.ceil(9.2) = %.1f%n", Math.ceil(a:9.2));
        System.out.printf(format:"Math.cos(0.0) = %.1f%n", Math.cos(a:0.0));
        System.out.printf(format:"Math.exp(1.0) = %.2f%n", Math.exp(a:1.0));
        System.out.printf(format:"Math.floor(9.2) = %.1f%n", Math.floor(a:9.2));
        System.out.printf(format:"Math.log(Math.E) = %.1f%n", Math.log(Math.E));
        System.out.printf(format:"Math.max(2.3, 12.7) = %.1f%n", Math.max(a:2.3, b:12.7));
        System.out.printf(format:"Math.min(2.3, 12.7) = %.1f%n", Math.min(a:2.3, b:12.7));
        System.out.printf(format:"Math.pow(2.0, 7.0) = %.1f%n", Math.pow(a:2.0, b:7.0));
        System.out.printf(format:"Math.sin(0.0) = %.1f%n", Math.sin(a:0.0));
        System.out.printf(format:"Math.sqrt(900.0) = %.1f%n", Math.sqrt(a:900.0));
        System.out.printf(format:"Math.tan(0.0) = %.1f%n", Math.tan(a:0.0));
        System.out.printf(format:"Math.random() = %.4f%n", Math.random()); // random each run
    }
}
```

Output:



```
Math.log(Math.E) = 1.0
Math.max(2.3, 12.7) = 12.7
Math.min(2.3, 12.7) = 2.3
Math.pow(2.0, 7.0) = 128.0
Math.sin(0.0) = 0.0
Math.sqrt(900.0) = 30.0
Math.tan(0.0) = 0.0
Math.random() = 0.1659
```

## 6.4 Give the method header for each of the following methods:

**a) Method hypotenuse, which takes two double-precision, floating-point arguments side1 and side2 and returns a double-precision, floating-point result.**

**b) Method smallest, which takes three integers x, y and z and returns an integer.**

**c) Method instructions, which does not take any arguments and does not return a value. [Note: Such methods are commonly used to display instructions to a user.]**

**d) Method intToFloat, which takes integer argument number and returns a float.**

Solution:

**a)** hypotenuse → takes two doubles, returns a double

```
public static double hypotenuse(double side1, double side2);
```

**b)** smallest → takes three integers, returns an integer

```
public static int smallest(int x, int y, int z);
```

**c)** instructions → no parameters, no return value (void)

```
public static void instructions();
```

**d)** intToFloat → takes an int, returns a float

```
public static float intToFloat(int number);
```

6.5 Find the error in each of the following program segments. Explain how to correct the error.

```
a) void g(){
        System.out.println("Inside method g");
        void h() {
        System.out.println("Inside method h");
         }
        }
```

```
b) int sum(int x, int y) {
        int result; result = x + y;
        }
c) void f(float a); {
       float a; System.out.println(a);
  }
d) void product() {
        int a = 6, b = 5, c = 4, result;
       result = a * b * c;
        System.out.printf("Result is %d%n", result);
       return result;
}
```

Solution:

a)

```
void g()
{
  System.out.println("Inside method g");
  void h()
  {
    System.out.println("Inside method h");
  }
}
```

Error: We cannot define a method inside another method. h() is declared inside g().

Fix: Move h() outside g().

```
void g() {
   System.out.println("Inside method g");
}


void h() {
   System.out.println("Inside method h");
}
```

b)

```
int sum(int x, int y)
{
  int result;
  result = x + y;
}
```

Error: Method declared to return int, but no return statement.

Fix: Add return result;.

```
int sum(int x, int y) {
   int result = x + y;
   return result;
}
```

c)

```
void f(float a);
{
  float a;
  System.out.println(a);
}
```

Errors:

1. The semicolon ; after void f(float a) is wrong → it ends the method header.

2. Variable float a; redeclares parameter a.

Fix: Remove semicolon, remove redeclaration.

```
void f(float a) {
   System.out.println(a);
}
```

d)

```
void product()
{
  int a = 6, b = 5, c = 4, result;
  result = a * b * c;
  System.out.printf("Result is %d%n", result);
  return result;
}
```

Error: Method declared void but trying to return result;.

Fix 1: If you want a return value → change method to int product().

Fix 2: If you don't want return → remove return result;.

Example with return:

```
int product() {
   int a = 6, b = 5, c = 4;
   int result = a * b * c;
   System.out.printf("Result is %d%n", result);
   return result;
}
```

Example without return:

```
void product() {
   int a = 6, b = 5, c = 4;
   int result = a * b * c;
   System.out.printf("Result is %d%n", result);
}
```

**6.6 <u>Declare method sphereVolume to calculate and return the volume of the sphere. Use the following statement to calculate the volume:</u>**

**<u>double volume = (4.0 / 3.0) * Math.PI * Math.pow(radius, 3)</u>**

**<u>Write a Java application that prompts the user for the double radius of a sphere, calls sphereVolume to calculate the volume and displays the result.</u>**

**<u>Solution:</u>**

Code:

```
import java.util.Scanner;
public class SphereVolumeCalculator {
   public static double sphereVolume(double radius) {
      double volume = (4.0 / 3.0) * Math.PI * Math.pow(radius, 3);
      return volume;
   }
   public static void main(String[] args) {
      Scanner input = new Scanner(System.in);
      System.out.print("Enter the radius of the sphere: ");
      double radius = input.nextDouble();
```

```
        double volume = sphereVolume(radius);
        System.out.printf("The volume of a sphere with radius %.2f is %.4f%n",
                radius, volume);
    }
}
```

Output:

```
Enter the radius of the sphere: 3.5
The volume of a sphere with radius 3.50 is 179.5944
```

6.7 What is the value of x after each of the following statements is executed?

a) x = Math.abs(-7.5);

b) x = Math.floor(5 + 2.5);

c) x = Math.abs(9) + Math.ceil(2.2);

d) x = Math.ceil(-5.2);

e) x = Math.abs(-5) + Math.abs(4);

f) x = Math.ceil(-6.4) - Math.floor(5.2);

g) x = Math.ceil(-Math.abs(-3 + Math.floor(-2.5)));

Solution:

**a)**

x = Math.abs(-7.5);

- abs(-7.5) = 7.5
  **x = 7.5**

**b)**

x = Math.floor(5 + 2.5);

- 5 + 2.5 = 7.5
- floor(7.5) = 7.0
  **x = 7.0**

**c)**

x = Math.abs(9) + Math.ceil(2.2);

- abs(9) = 9
- ceil(2.2) = 3.0
- 9 + 3.0 = 12.0
  **x = 12.0**

**d)**

x = Math.ceil(-5.2);

- ceil(-5.2) = the smallest integer **≥ -5.2** → -5.0
  **x = -5.0**

**e)**

x = Math.abs(-5) + Math.abs(4);

- abs(-5) = 5
- abs(4) = 4
- 5 + 4 = 9
  **x = 9**

**f)**

x = Math.ceil(-6.4) - Math.floor(5.2);

- ceil(-6.4) = smallest integer ≥ -6.4 → -6.0
- floor(5.2) = largest integer ≤ 5.2 → 5.0
- -6.0 - 5.0 = -11.0
  **x = -11.0**

**g)**

x = Math.ceil(-Math.abs(-3 + Math.floor(-2.5)));

Step by step:

- floor(-2.5) = -3.0
- -3 + (-3.0) = -6.0
- Math.abs(-6.0) = 6.0
- -Math.abs(...) = -6.0
- ceil(-6.0) = -6.0
  **x = -6.0**

## Section 4.2

**(a)**

**Question:**
Evaluate the following method call:

Math.sqrt(4)

**Answer:**
$\sqrt{4}$ = **2.0**

**(b)**

**Question:**
Evaluate the following method call:

Math.sin(2 * Math.PI)

**Answer:**
sin(2π) = **0.0**

**(c)**

**Question:**
Evaluate the following method call:

Math.cos(2 * Math.PI)

**Answer:**
cos(2π) = **1.0**

**(d)**

**Question:**
Evaluate the following method call:

Math.pow(2, 2)

**Answer:**
$2^2$ = **4.0**

**(e)**

**Question:**
Evaluate the following method call:

Math.log(Math.E)

**Answer:**
$\log_e(e)$ = **1.0**

**(f)**

**Question:**
Evaluate the following method call:

Math.exp(1)

**Answer:**
$e^1$ = **2.718281828...**

**(g)**

**Question:**
Evaluate the following method call:

Math.max(2, Math.min(3, 4))

**Answer:**
min(3,4)=3 → max(2,3)= **3**

**(h)**

**Question:**
Evaluate the following method call:

Math.rint(-2.5)

**Answer:**
Nearest even integer = **-2.0**

**(i)**

**Question:**
Evaluate the following method call:

Math.ceil(-2.5)

**Answer:**
Smallest integer ≥ -2.5 = **-2.0**

**(j)**

**Question:**
Evaluate the following method call:

Math.floor(-2.5)

**Answer:**
Largest integer ≤ -2.5 = **-3.0**

**(k)**

**Question:**
Evaluate the following method call:

Math.round(-2.5f)

**Answer:**
Nearest int = **-2**

**(l)**

**Question:**
Evaluate the following method call:

Math.round(-2.5)

**Answer:**
Nearest int = **-2**

**(m)**

**Question:**
Evaluate the following method call:

Math.rint(2.5)

**Answer:**
Nearest even integer = **2.0**

**(n)**

**Question:**
Evaluate the following method call:

Math.ceil(2.5)

**Answer:**
Smallest integer ≥ 2.5 = **3.0**

**(o)**

**Question:**
Evaluate the following method call:

Math.floor(2.5)

**Answer:**
Largest integer ≤ 2.5 = **2.0**

**(p)**

**Question:**
Evaluate the following method call:

Math.round(2.5f)

**Answer:**
Nearest int = **3**

**(q)**

**Question:**
Evaluate the following method call:

Math.round(2.5)

**Answer:**
Nearest int = **3**

**(r)**

**Question:**
Evaluate the following method call:

Math.round(Math.abs(-2.5))

**Answer:**
abs(-2.5)=2.5 → round(2.5)= **3**

4.2.2

Question:
True or false? The argument for trigonometric methods is an angle in radians.

Answer:
True – The argument for all trigonometric methods (Math.sin(), Math.cos(), etc.) is in radians.

4.2.3

Question:
Write a statement that converts 47 degrees to radians and assigns the result to a variable.

Answer:

```
double rad = Math.toRadians(47);
```

.2.4

Question:
Write a statement that converts π / 7 to an angle in degrees and assigns the result to a variable.

Answer:

```
double deg = Math.toDegrees(Math.PI / 7);
```

4.2.5

Question:
Write an expression that obtains:

1. A random integer between 34 and 55.
2. A random integer between 0 and 999.
3. A random number between 5.5 and 55.5.

Answer:

```
int num1 = 34 + (int)(Math.random() * (55 - 34 + 1));  // 34–55
int num2 = (int)(Math.random() * 1000);             // 0–999

double num3 = 5.5 + Math.random() * (55.5 - 5.5);     // 5.5–55.5
```

4.2.6

Question:
Why does the Math class not need to be imported?

Answer:
Because the Math class is in the java.lang package, which is automatically imported in every Java program.

4.2.7

Question:
What is:

1. Math.log(Math.exp(5.5))
2. Math.exp(Math.log(5.5))
3. Math.asin(Math.sin(Math.PI / 6))
4. Math.sin(Math.asin(Math.PI / 6))

Answer:

1. Math.log(Math.exp(5.5)) → 5.5
2. Math.exp(Math.log(5.5)) → 5.5
3. Math.asin(Math.sin(Math.PI / 6)) → π/6 ≈ 0.523598...
4. Math.sin(Math.asin(Math.PI / 6)) → π/6 ≈ 0.523598...

## **<u>Section 4.3</u>**

4.3.1

Question:
Use print statements to find out the ASCII code for '1', 'A', 'B', 'a', and 'b'.
Use print statements to find out the character for the decimal codes 40, 59, 79, 85,

and 90.
Use print statements to find out the character for the hexadecimal codes 40, 5A, 71, 72, and 7A.

Answer:

```
public class Test {

  public static void main(String[] args) {

    // ASCII codes

    System.out.println((int)'1'); // 49

    System.out.println((int)'A'); // 65

    System.out.println((int)'B'); // 66

    System.out.println((int)'a'); // 97

    System.out.println((int)'b'); // 98


    // Decimal codes → characters

    System.out.println((char)40); // (

    System.out.println((char)59); // ;

    System.out.println((char)79); // O

    System.out.println((char)85); // U

    System.out.println((char)90); // Z


    // Hexadecimal codes → characters

    System.out.println((char)0x40); // @

    System.out.println((char)0x5A); // Z

    System.out.println((char)0x71); // q

    System.out.println((char)0x72); // r

    System.out.println((char)0x7A); // z

  }
```

}

4.3.2

Question:
Which of the following are correct literals for characters?
'1', '\u345dE', '\u3fFa', '\b', '\t'

Answer:
Correct literals:

- '1' (valid character literal)
- '\b' (backspace)
- '\t' (tab)

Incorrect:

- '\u345dE' → invalid (too many characters)
- '\u3fFa' → invalid (too many characters)

4.3.3

Question:
How do you display the characters \ and "?

Answer:

```
System.out.println("\\");  // prints \

System.out.println("\"");  // prints "
```

4.3.4

Question:
Evaluate the following:

```
int i = '1';

int j = '1' + '2' * ('4' - '3') + 'b' / 'a';

int k = 'a';
```

char c = 90;

Answer:

- int i = '1'; → '1' ASCII = 49
- int j = '1' + '2' * ('4' - '3') + 'b' / 'a';
    - '2' * ('4' - '3') → 50 * 1 = 50
    - 'b' / 'a' → 98 / 97 = 1
    - '1' + 50 + 1 → 49 + 50 + 1 = 100
- int k = 'a'; → 'a' ASCII = 97
- char c = 90; → ASCII 90 = 'Z'

4.3.5

Question:
Can the following conversions involving casting be allowed? If so, find the converted result.

char c = 'A';

int i = (int)c;

float f = 1000.34f;

int i = (int)f;

double d = 1000.34;

int i = (int)d;

int i = 97;

char c = (char)i;

Answer:

- char c = 'A'; int i = (int)c; → i = 65

- float f = 1000.34f; int i = (int)f; → i = 1000
- double d = 1000.34; int i = (int)d; → i = 1000
- int i = 97; char c = (char)i; → c = 'a'

4.3.6

Question:
Show the output of the following program:

```
public class Test {
  public static void main(String[] args) {
    char x = 'a';
    char y = 'c';

    System.out.println(++x);
    System.out.println(y++);
    System.out.println(x - y);
  }
}
```

Answer:

- ++x → 'a' becomes 'b' → prints b
- y++ → prints 'c', then y becomes 'd'
- x - y → 'b' - 'd' → 98 - 100 = -2

Output:

```
b
c
-2
```

4.3.7

Question:
Write the code that generates a random lowercase letter.

Answer:

```
char ch = (char)('a' + Math.random() * 26);
```

4.3.8

Question:
Show the output of the following statements:

```
System.out.println('a' < 'b');

System.out.println('a' <= 'A');

System.out.println('a' > 'b');

System.out.println('a' >= 'A');

System.out.println('a' == 'a');

System.out.println('a' != 'b');
```

Answer:

- 'a' < 'b' → true
- 'a' <= 'A' → false
- 'a' > 'b' → false
- 'a' >= 'A' → true
- 'a' == 'a' → true
- 'a' != 'b' → true

## Section 4.4

4.4.1

Question:
Suppose that s1, s2, and s3 are three strings, given as follows:

String s1 = "Welcome to Java";

String s2 = "Programming is fun";

String s3 = "Welcome to Java";

What are the results of the following expressions?

a. s1 == s2
b. s2 == s3
c. s1.equals(s2)
d. s1.equals(s3)
e. s1.compareTo(s2)
f. s2.compareTo(s3)
g. s2.compareTo(s2)
h. s1.charAt(0)
i. s1.indexOf('j')
j. s1.indexOf("to")
k. s1.lastIndexOf('a')
l. s1.lastIndexOf("o", 15)
m. s1.length()
n. s1.substring(5)
o. s1.substring(5, 11)
p. s1.startsWith("Wel")
q. s1.endsWith("Java")
r. s1.toLowerCase()
s. s1.toUpperCase()
t. s1.concat(s2)
u. s1.contains(s2)
v. "\t Wel \t".trim()

Answer:
a. false
b. false
c. false

d. true
e. negative value (since "Welcome..." < "Programming..." lexicographically → about 7)
f. positive value
g. 0
h. 'W'
i. -1 (no lowercase j in s1)
j. 8
k. 14
l. 11
m. 15
n. "me to Java"
o. "me to "
p. true
q. true
r. "welcome to java"
s. "WELCOME TO JAVA"
t. "Welcome to JavaProgramming is fun"
u. false
v. "Wel"

4.4.2

Question:
Suppose that s1 and s2 are two strings. Which of the following statements or expressions are incorrect?

String s = "Welcome to Java";

String s3 = s1 + s2;

String s3 = s1 s2;

s1 == s2;

s1 >= s2;

s1.compareTo(s2);

int i = s1.length();

char c = s1(0);

char c = s1.charAt(s1.length());

Answer:
Incorrect statements:

- String s3 = s1 s2; → missing + operator
- s1 >= s2; → relational operators (>, <, >=, <=) not allowed for strings
- char c = s1(0); → should be s1.charAt(0)
- char c = s1.charAt(s1.length()); → out of bounds (index should be 0 to length-1)

Correct statements:

- String s = "Welcome to Java";
- String s3 = s1 + s2;
- s1 == s2; (compiles, but compares references)
- s1.compareTo(s2);
- int i = s1.length();

4.4.3

Question:
Show the output of the following statements:

System.out.println("1" + 1);

System.out.println('1' + 1);

System.out.println("1" + 1 + 1);

System.out.println("1" + (1 + 1));

System.out.println('1' + 1 + 1);

Answer:

11

50

111

12

51

4.4.4

Question:
Evaluate the following expressions:

1 + "Welcome " + 1 + 1

1 + "Welcome " + (1 + 1)

1 + "Welcome " + ('\u0001' + 1)

1 + "Welcome " + 'a' + 1

Answer:

1Welcome 11

1Welcome 2

1Welcome 2

1Welcome a1

4.4.5

Question:
Let s1 = " Welcome " and s2 = " welcome ". Write code for the following:

Answer:

boolean isEqual;

int x;

boolean b;

String s3;

char c;

// a

```
isEqual = s1.equals(s2);

// b
isEqual = s1.equalsIgnoreCase(s2);

// c
x = s1.compareTo(s2);

// d
x = s1.compareToIgnoreCase(s2);

// e
b = s1.startsWith("AAA");

// f
b = s1.endsWith("AAA");

// g
x = s1.length();

// h
c = s1.charAt(0);

// i
s3 = s1 + s2;
```

```
// j
s3 = s1.substring(1);

// k
s3 = s1.substring(1, 4);

// l
s3 = s1.toLowerCase();

// m
s3 = s1.toUpperCase();

// n
s3 = s1.trim();

// o
x = s1.indexOf('e');

// p
x = s1.lastIndexOf("abc");
```

4.4.6

Question:
Write one statement to return the number of digits in an integer i.

Answer:

```
int digits = Integer.toString(Math.abs(i)).length();
```

4.4.7

Question:
Write one statement to return the number of digits in a double value d.

Answer:

```
int digits = Double.toString(Math.abs(d)).replace(".", "").length();
```

4.4.8

Question:
What is wrong in the following code?

```
import java.util.Scanner;


public class Test {

  public static void main(String[] args) {

    Scanner input = new Scanner(System.in);

    System.out.print("Enter an integer: ");

    int value = input.nextInt();

    System.out.println("The value is " + value);


    System.out.print("Enter a line: ");

    String line = input.nextLine();

    System.out.println("The line is " + line);

  }

}
```

Answer:
The second nextLine() reads the leftover newline from nextInt(), so line becomes empty.
Fix: add input.nextLine(); after reading the integer:

```
int value = input.nextInt();

input.nextLine(); // consume newline

String line = input.nextLine();
```

## **Section 4.5**

4.5.1

Question:
If you run GuessBirthday.java with input 1 for Set1, Set3, and Set4 and 0 for Set2 and Set5, what will be the birthday?

Answer:
Birthday = 29

*(Because the program adds the first number of each "Yes" set: 1 + 4 + 8 + 16 = 29)*

4.5.2

Question:
If you enter a lowercase letter such as b, the program in Listing 4.4 displays B is 11. Revise the code to display b is 11.

Answer:
Use Character.toLowerCase(ch) when printing:

```
System.out.println(Character.toLowerCase(ch) + " is " + number);
```

4.5.3

Question:
What would be wrong if lines 6-7 in Listing 4.5 are replaced by:

```
String lottery = "" + (int)(Math.random() * 100);
```

Answer:
This produces a number from 0 to 99, not a 2-digit lottery from 00 to 99, so numbers like 5 will be "5" instead of "05".

## **Section 4.6**

4.6.1

Question:
What are the format specifiers for outputting a Boolean value, a character, a decimal integer, a floating-point number, and a string?

Answer:

- Boolean → %b
- Character → %c
- Decimal integer → %d
- Floating-point → %f
- String → %s

4.6.2

Question:
What is wrong in the following statements?

a. System.out.printf("%5d %d\n", 1, 2, 3); → Too many arguments
b. System.out.printf("%5d %f\n", 1); → Missing argument for %f
c. System.out.printf("%5d %f\n", 1, 2); → 2 is int, should be double for %f
d. System.out.printf("%.2f\n%0.3f\n", 1.23456, 2.34); → Correct
e. System.out.printf("%08s\n", "Java"); → %0 ignored for string, prints " Java"

4.6.3

Question:
Show the output of the following statements:

(a) System.out.printf("amount is %f %e\n", 32.32, 32.32);

(b) System.out.printf("amount is %5.2f%% %5.4e\n", 32.327, 32.32);

(c) System.out.printf("%6b\n", (1 > 2));

(d) System.out.printf("%6s\n", "Java");

(e) System.out.printf("%-6b%s\n", (1 > 2), "Java");

(f) System.out.printf("%6b%-8s\n", (1 > 2), "Java");

(g) System.out.printf("%,5d %,6.1f\n", 312342, 315562.932);

(h) System.out.printf("%05d %06.1f\n", 32, 32.32);

Answer:

(a) amount is 32.320000 3.232000e+01

(b) amount is 32.33% 3.2320e+01

(c)  false

(d)   Java

(e) false Java

(f)  falseJava

(g) 312,342 315,562.9

(h) 00032 032.3