PATUAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

COURSE CODE CCE-121

SUBMITTED TO:

Prof. Dr. Md Samsuzzaman

Department of Computer and Communication Engineering
Faculty of Computer Science and Engineering

SUBMITTED BY:

Md. Sadman Kabir Bhuiyan

ID: **2102020**,

Registration No: **10147**

Faculty of Computer Science and Engineering

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 **last in, first out (LIFO)** 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**, **return expression** and **encountering the last closing right brace**.

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 **stack frame** or **activation record** 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) **stackoverflow** 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.

Class body

b) the variable die1.

block that defines method rollDice's body.

c) the method rollDice.

Class body

d) the method main.

Class body

e) the variable sumOfDice.

block that defines method main's body.

6.3 Write an application that tests whether the examples of the Math class method calls shown in Fig. 6.2 actually produce the

indicated results.

```
public class MathTest
{
public static void main(String[] args)
{
System.out.printf("Math.abs(23.7) = %f/n", Math.abs(23.7));
System.out.printf("Math.abs(0.0) = %f/n", Math.abs(0.0));
System.out.printf("Math.abs(-23.7) = %f/n ", Math.abs(-23.7));
System.out.printf("Math.ceil(9.2) = %f/n ", Math.ceil(9.2));
System.out.printf("Math.ceil(-9.8) = %f/n ", Math.ceil(-9.8));
System.out.printf("Math.cos(0.0) = %f/n ", Math.cos(0.0));
System.out.printf("Math.exp(1.0) = %f/n ", Math.exp(1.0));
System.out.printf("Math.exp(2.0) = %f/n ", Math.exp(2.0));
System.out.printf("Math.floor(9.2) = %f/n", Math.floor(9.2));
System.out.printf("Math.floor(-9.8) = %f/n ", Math.floor(-9.8));
System.out.printf("Math.log(Math.E) = %f/n ", Math.log(Math.E));
System.out.printf("Math.log(Math.E * Math.E) = %f/n ",
Math.log(Math.E * Math.E));
System.out.printf("Math.max(2.3, 12.7) = %f/n ", Math.max(2.3, 12.7));
System.out.printf("Math.max(-2.3, -12.7) %f/n n",
Math.max(-2.3, -12.7));
System.out.printf("Math.min(2.3, 12.7) = %f/n ", Math.min(2.3, 12.7));
System.out.printf("Math.min(-2.3, -12.7) = %f/n",
Math.min(-2.3, -12.7));
System.out.printf("Math.pow(2.0, 7.0) = %f/n ", Math.pow(2.0, 7.0));
System.out.printf("Math.pow(9.0, 0.5) = %f/n ", Math.pow(9.0, 0.5));
System.out.printf("Math.sin(0.0) = %f/n”Math.sin(0.0));
System.out.printf("Math.sqrt(900.0) = %f/n ", Math.sqrt(900.0));
System.out.printf("Math.tan(0.0) = %f/n ", Math.tan(0.0));
}
}
```

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.

```
double hypotenuse(double side1, double side2)
```

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

```
int smallest(int x, int y, int z)
```

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.]

```
void instructions()
```

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

```
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");
}
}
```

Error: Method h is declared within method g.

Correction: Move the declaration of h outside the declaration of g.

b)

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

Error: The method is supposed to return an integer, but does not.

Correction: Delete the variable result, and place the statement

```
return x + y;
```

in the method, or add the following statement at the end of the method body:

```
return result;
```

c)

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

Error: The semicolon after the right parenthesis of the parameter list is incorrect, and

the parameter a should not be redeclared in the method.

Correction: Delete the semicolon after the right parenthesis of the parameter list, and

delete the declaration float 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: The method returns a value when it's not supposed to.

Correction: Change the return type from void to int.

6.6 Declare method sphereVolume to calculate and return the volume of the sphere. Use the following statement to calculate the volume:

```
import java.util.Scanner;
public class Sphere
{
// obtain radius from user and display volume of sphere
public static void main(String[] args)
{
Scanner input = new Scanner(System.in);
System.out.print("Enter radius of sphere: ");
double radius = input.nextDouble();
System.out.printf("Volume is %f%n", sphereVolume(radius));
} // end method determineSphereVolume
// calculate and return sphere volume
public static double sphereVolume(double radius)
{
double volume = (4.0 / 3.0) * Math.PI * Math.pow(radius, 3);
return volume;
} // end method sphereVolume
} // end
```

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

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

> 7.5

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

> 7.0

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

> 12.0

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

> -5.0

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

```
> 9.0
f) x = Math.ceil(-6.4) - Math.floor(5.2);
> -11.0
g) x = Math.ceil(-Math.abs(-3 + Math.floor(-2.5)));
> -6.0
```

6.8 (Parking Charges)

```java
import java.util.Scanner;

class Solution
{
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
int hours_parked = scanner.nextInt();
scanner.close();

double fee = calculateParkingFee(hours_parked);
System.out.println(hours_parked + " hours parked = " + fee);
}

public static double calculateParkingFee(int hours_parked) {
double fee = 0.0;
if (hours_parked <= 3) {
fee = 2.00;
} else if (hours_parked <= 19) {
fee = 2.00 + 0.50 * (hours_parked - 3);
} else {
fee = 10.00;
}
return fee;
}
}
```

6.9 (Rounding Numbers)

```java
import java.util.Scanner;

class Solution
{
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
double x = scanner.nextDouble();
scanner.close();

double y = Math.floor(x + 0.5);
System.out.println("Original: " + x + ", Rounded: " + y);
}
}
```

6.10 (Rounding Numbers)

```java
import java.util.Scanner;

public class Solution
{
public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
double x = scanner.nextDouble();
scanner.close();

System.out.println("Original: " + x + "\n" +
"Round to Integer: " + roundToInteger(x) + "\n" +
"Round to Tenths: " + roundToTenths(x) + "\n" +
"Round to Hundredths: " + roundToHundredths(x) + "\n" +
"Round to Thousandths: " + roundToThousandths(x));
}

public static double roundToInteger(double x) {
return Math.floor(x + 0.5);
}

public static double roundToTenths(double x) {
return Math.floor(x * 10 + 0.5) / 10;
}

public static double roundToHundredths(double x) {
return Math.floor(x * 100 + 0.5) / 100;
}

public static double roundToThousandths(double x) {
return Math.floor(x * 1000 + 0.5) / 1000;
}
}
```

6.11 Answer each of the following questions:

a) What does it mean to choose numbers “at random”?

Ans: Choosing at random means to choose a integer or floating point number without learning it's value randomly.

b) Why is the nextInt method of class SecureRandom useful for simulating games of chance?

Ans: Random can be useful to develop neutral mobs and mods in games, like in a car racing game, the direction of other bot cars.

c) Why is it often necessary to scale or shift the values produced by a SecureRandom object?

Ans: Scalling or shifting may be necessary to get the output in between a number or a range.

d) Why is computerized simulation of real-world situations a useful technique?

Ans: With simulations we can observe real world situations and other complex systems, which enables researchers to learn more about a system.

6.12 Write statements that assign random integers to the variable n in the following ranges:

a) $2 \leq n \leq 6$.

```
> int n = random.nextInt(5) + 2;
```

b) $4 \leq n \leq 50$.

```
> int n = random.nextInt(47) + 4;
```

c) $0 \leq n \leq 7$.

```
> int n = random.nextInt(8);
```

d) $1000 \leq n \leq 1030$.

```
> int n = random.nextInt(31) + 1000;
```

e) $-5 \leq n \leq 1$.

```
> int n = random.nextInt(7) - 5;
```

f) $-2 \leq n \leq 9$.

```
> int n = random.nextInt(12) - 2;
```

6.13 Write statements that will display a random number from each of the following sets:

a) 0, 3, 6, 9, 12.

```
n = (int) (Math.random() * 5) * 3;
```

b) 1, 2, 4, 8, 16, 32.

```
n = (int) Math.pow(2, (int) (Math.random() * 6));
```

c) 10, 20, 30, 40.

```
n = (int) (Math.random() * 4 + 1) * 10;
```

6.14 (Floor and Ceil)

```
public class Round {
```

```java
public static void main(String[] args) {
Round round = new Round();

System.out.println(round.myFloor(3.5));
System.out.println(round.myFloor(-3.5));
System.out.println(round.myCeil(-3.5));
System.out.println(round.myCeil(-3.6));
}

double myFloor(double x) {
return Math.floor(x);
}

double myCeil(double x) {
return Math.ceil(x);
}
}
```

6.15 (Hypotenuse Calculations)

Ans:

```java
public class Triangle {
public static void main(String[] args) {
double side1, side2, hypotenuse;
side1 = 3.0;
side2 = 4.0;
hypotenuse = hypotenuse(side1, side2);
System.out.println("Given sides of lengths " + side1 + " and " + side2 + " the hypotenuse is " + hypotenuse);

side1 = 5.0;
side2 = 12.0;
hypotenuse = hypotenuse(side1, side2);
System.out.println("Given sides of lengths " + side1 + " and " + side2 + " the hypotenuse is " + hypotenuse);

side1 = 8.0;
side2 = 15.0;
hypotenuse = hypotenuse(side1, side2);
System.out.println("Given sides of lengths " + side1 + " and " + side2 + " the hypotenuse is " + hypotenuse);
}

static double hypotenuse(double a, double b) {
return Math.sqrt(a * a + b * b);
}
}
```