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. Sharafat Karim

ID: **2102024**,

Registration No: **10151**

Faculty of Computer Science and Engineering

Date of submission: **6 September, 2023**

Assignment: 01
Assignment based on CCE 121 Question

Assignment title: CCE 121 Question

1 [A] Fill in the blanks in each of the following statements:

i) If a class declares constructors, the compiler will not create a default constructor

ii) The public methods of a class are also known as the class's public interface

or API

iii) Lists and tables of values can be stored in arrays

and data structure.

iv) The number used to refer to a particular array element is called the elements index.

v) A variable known only within the method in which it's declared is called a local variable.

vi) 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.

vii) Typically, for statements are used for counter-controlled repetition and while statements for sentinel-controlled repetition.

viii) Methods that perform common tasks and do not require objects are called

methods static.

[B] Write a Java statement or a set of Java statements to accomplish each of the following tasks:

a) Sum the odd integers between 1 and 99, using a for statement. Assume that the integer variables sum and count have been declared.

```java
class Main {

public static void main(String args[]) {
int sum = 0, count = 0;
for (; sum < 100; sum++)
{
if (sum % 2 != 0)
{
count += sum;
}
}
System.out.println("Sum : " + count);
}
}
```

b) Print the integers from 1 to 20, using a while loop and the counter variable i. Assume that the variable i has been declared, but not initialized. Print only five integers per line.

```java
class Main {

public static void main(String args[]) {
int i = 1;
while (i <= 20) {
System.out.println(i++ + " " + i++ + " " + i++ + " " + i++ + " " + i++);
}
}
}
```

c) Repeat part (b), using a for statement.

```
class Main {

public static void main(String args[]) {
int i = 1;
for (; i <= 20; i++) {
System.out.println(i++ + " " + i++ + " " + i++ + " " + i++ + " " + i);
}
}
}
```

[C] i) What gives Java its 'write once and run anywhere' nature?

Ans: Java has JRE which let’s it’s bytecode cross platform alongside the ability to run anywhere.

ii) What happens at runtime during Java compilation?

Ans: During compilation Java compiler javac makes a bytecode from the source code.

iii) Can you save a Java source file by another name than the class name?

Ans: Yes, I can if there is no public class.

iv) Can you have multiple classes in a java source file?

Ans: Yes.

[D] Write a Java program to create and display unique three-digit number using 1, 2, 3, 4. Also count
how many three-digit numbers are there.

```
class Main {

public static void main(String args[]) {
int i = 1, j = 1, k = 1, count = 0;
for (i=1; i <= 4; i++)
{
for (j=1; j <=4; j++)
{
for (k=1; k <= 4; k++)
{
if (i!=j && k!=j && i!=k )
{
System.out.println(i+""+j+""+k);
count++;
```

```java
}
}
}
}
System.out.println("Total found: " + count);
}
}
```

Output:

```
123

124

132

134

142

143

213

214

231

234

241

243

312

314

321

324

341

342

412

413

421

423

431

432

Total found: 24
```

2 [A] What are the various access specifiers in Java? Write an example of public access modifier.

Ans: There are four kinds of access specifiers, public, protected, private and default.

One public access modifier example is,

```
class Main {

public static void main(String args[]) {
System.out.println("Hello java");
}
}

```

[B] Write the rules of Constructor. What is the purpose of a default constructor? Explain with example.

Ans: In java constructor is like a method which executes once an instance of a class is called. It is a special kind of mehod.

Constructor is used to perform specific task according to the input in a object. One example is,

```
/**

 * InnerMain
 */
class InnerMain {

int value;
void printInner()
{
System.out.println("Value is: " + value);
 }
 }

class Main {
public static void main(String args[]) {
System.out.println("Hello java");
InnerMain obj = new InnerMain();
obj.printInner();
}
}


```

[C] output

(i) ans: a = 10, b = 15

(ii) ans: 0

(iii) ans: Compiler error.

[D]

```
import java.util.Scanner;

class Main {
public static void main(String args[]) {
int pyramidHeight;
Scanner sc = new Scanner(System.in);
pyramidHeight = sc.nextInt();

for (int i = 0; i < pyramidHeight; i++) {
 for (int j = 0; j < pyramidHeight - i - 1; j++) {
 System.out.print(" ");
 }
 for (int k = 0; k < i + 1; k++) {
 System.out.print("*");
 }
 for (int l = 0; l < i; l++) {
 System.out.print("*");
 }
 System.out.println();
 }
 }
 }

```

3 [A] What is the static variable? Explain a java program with and without static variable.

Ans: Static variable is a type variable which share it's value among all instances of classes. Memory allocation happens only once.

```
class Main {

static int instanceCount = 0;
public static void main(String args[]) {
instanceCount++;
System.out.println("Instance count: " + instanceCount);
}
}

```

without static variable,

```
class Main {

public static void main(String args[]) {
int instanceCount = 1;
instanceCount++;
System.out.println("Instance count: " + instanceCount);
}
}

```

[B] i) What is the difference between static (class) method and instance method?

Ans: The main difference between static and instance methods is that static methods are associated with the class itself, while instance methods are associated with each object of the class. This means that static methods can be called without creating an object of the class, while instance methods must be called on an object.

ii) What are the main uses of this keyword?

Ans: static keyword is used for static method and static variables.

[C] **Define Object and Class. Write Object and Class Example: main outside the class and main within the class**

Ans: Define Object and Class. Write Object and Class Example: main outside the class and main within the class.

```
class Main {

public static void main(String args[]) {
int instanceCount = 1;
instanceCount++;
System.out.println("Instance count: " + instanceCount);
}
}

```

and outside the main class,

```
class value {

String a;
}
class Main {
public static void main(String args[]) {
value foo = new value();
foo.a = "Hello";
System.out.println(foo.a);
}
}
```

[D] Write a Java program to sort an array of given integers using the Bubble sorting Algorithm
Original Array:[7, -5, 3, 2, 1, 0, 45]
Sorted Array : [-5, 0, 1, 2, 3, 7, 45]

```
class Main {
public static void main(String args[]) {
int arr[] = { 7, -5, 3, 2, 1, 0, 45 };
int n = arr.length;
for (int i = 0; i < n - 1; i++) {
for (int j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
int temp = arr[j];
arr[j] = arr[j + 1];
arr[j + 1] = temp;
}
}
}
System.out.println("Sorted array: ");
for (int i = 0; i < n; i++) {
System.out.println(arr[i] + " ");
}
}
}
```

4 [A] Differentiate between the throw and throws keyword.

Throw keyword is used to explicitly say that a method can throw an exception. And on the other hand throws keyword is used to say that a method can throw such type exception.

[B] “Aggregation represents HAS-A relationship.”-explain with example.

Aggregation is a type of relationship between two classes in object-oriented programming (OOP). It is a **has-a** relationship, meaning that one class has an instance of another class. The class that has the instance is called the **aggregate**, and the class that is being instantiated is called the **part**.

For example, a `Car` class can have an aggregation relationship with a `Engine` class. This means that a `Car` object can have an `Engine` object. The `Car` object is the aggregate, and the `Engine` object is the part.

In this relationship, the `Engine` object is independent of the `Car` object. This means that the `Engine` object can be used in other classes, such as a `Truck` class.

[C] Is it possible to make any class read-only or write-only in java? How?

Ans: It is possible to make any class read-only or write-only in Java. There are two ways to do this:

1. **Make all of the data members private.** This will prevent anyone from accessing the data members directly. If you need to access the data members, you can create getter and setter methods.
2. **Do not provide any setter methods for the data members.** This will prevent anyone from modifying the data members. However, you can still access the data members through the getter methods.

[D] What is the use of instance initializer block while we can directly assign a value in instance datamember?
Ans: An instance initializer block is a block of code that is executed when an instance of a class is created. It is used to initialize the instance data members of the class.
The instance initializer block is similar to a constructor, but there are some key differences. First, the instance initializer block is executed before the constructor. Second, the instance initializer block cannot be used to call other methods.

[E] How can you achieve abstraction in java?

Ans: Abstraction in Java can be achieved using either abstract classes or interfaces.

- **Abstract classes** are classes that can have abstract methods. Abstract methods are methods that do not have a body. The body of the abstract method is implemented in the subclasses of the abstract class.
- **Interfaces** are blueprints of classes. They define the methods that a class must implement, but they do not define the implementation of the methods.

6. [A] Output of a java program

(I)

```java
class Dog{
public static void main(String[] args) {
Dog d = null;
System.out.println(d instanceof Dog);
}
}
```

Ans: false

(ii)

```java
class A {
protected void msg() {
System.out.println("Hello java");
}
}

public class Simple extends A {
void msg() {
System.out.println("Hello java");
}

public static void main(String[] args) {
Simple obj = new Simple();
obj.msg();
}
}
```

Ans: Compiler error! But it’ll print “Hello java” if protected is removed from the method msg of class A.

(iii)

```
public class Sample {
public static void main(String[] args) {
try {
int data = 100 / 0;
}
catch (ArithmeticException e)
{
System.out.println(e);
}
System.out.println("rest of the code ... ");
}
}
```

Ans: java.lang.ArithmeticException: / by zero

rest of the code...

(iv)

```
class Animal {
void eat()
{
System.out.println("eating ... ");
}
}
class Dog extends Animal {
void bark()
{
System.out.println("barking ... ");
}
}
class Cat extends Animal {
void meow()
{
System.out.println("meowing ... ");
}
}
public class TestInheritance3 {
public static void main(String[] args) {
Cat c = new Cat();
c.meow();
c.eat();
c.bark();
}
}
```

Ans: Compiler error (bark method not available).

Without c.bark() it will print meowing… and eating…

6[B] How to access package from another package?

Ans: In java to access package from another one, we use the import command. For example,

    import packagename.classname;

6[C] What is the purpose of join method?

Ans: join method is used when one thread waits for an another thread to finish it's job. For example, if we want to finish the process of ThreadB and then continue, we can use,

    ThreadB.join();

6[D] How to perform two tasks by two threads?

Ans: To perform two task, we can two separate threads and assign each of them two different task and start them. In this way those two will start working simultaneously.

6[E] What is the Thread Scheduler and what is the difference between preemptive scheduling and time

slicing?

A component of Java that decides which thread to run or execute and which thread to wait is called a **thread scheduler in Java**. In Java, a thread is only chosen by a thread scheduler if it is in the runnable state. However, if there is more than one thread in the runnable state, it is up to the thread scheduler to pick one of the threads and ignore the other ones.

- **Preemptive scheduling** allows the scheduler to take a running thread away from the CPU and give it to another thread, even if the first thread has not finished its execution. This is done based on the priority of the threads. A thread with a higher priority will preempt a thread with a lower priority.
- **Time slicing** is a type of preemptive scheduling where each thread is given a certain amount of time to run on the CPU. After the time slice expires, the thread is preempted and another thread is given a chance to run. This process continues until all of the threads have had a chance to run.

The main difference between preemptive scheduling and time slicing is that in preemptive scheduling, the scheduler can preempt a running thread at any time, while in time slicing, the thread is only preempted after its time slice expires.