National University of Computer and Emerging Sciences, Lahore Campus

	Course Name:	Software Construction & Development	Section:	ALL
	Program:	BS (Software Engineering)	Semester:	Fall 2022
	Duration:	1 Hour	Total Marks:	40
	Evaluation Type:	Mid 1 Exam	Weight:	15 %
	Course Code:	SE3001	Page(s)	7
	Name:		Roll Number:	

Important Note:

- The quality of the code will affect the marks.
- Students will receive **ZERO** marks if the answers are plagiarized.
- Use of **mobile phones**, **internet**, and, **ANY type of smart devices** during the exam is strictly prohibited.
- Discussion with other students is not allowed.
- Exchange of notes and stationery with other students are not allowed.
- If any of the above rules is violated by the student. The invigilator has the right to file DC case against that student and the invigilator also has the right to take your exam away and ask you to leave the exam hall.

Question 1[CLO1]: Tracing

a) **[4 marks]** For each part, either check the box that indicates that a compiler error would occur, or write the output on the line associated with the code fragment.

```
public class X {
private void foo() {
System.out.print( "A" );
}
public void bar() {
System.out.print( "B" );
foo();
}
public static void main( String []
args ) {
X x = new X();
System.out.print( "C" );
x.bar();
}
}
```

Compiler Error: □

OR

Write Output:

```
public class X {
private void foo() {
System.out.print( "A" );
}
public void bar() {
System.out.print( "B" );
foo();
}
}
public class Y {
public void banana() {
X x = new X();
System.out.print( "C" );
x.foo();
}
public static void main( String []
args ) {
Y y = new Y();
System.out.print( "C" );
y.banana();
}}
```

Compiler Error: □

OR

Write Output:

```
public class X {
private int orange = 0;
public void bar() {
orange++;
System.out.print( "B" + orange );
}
public static void main( String []
args ) {
X x = new X();
System.out.print( "C" );
x.bar();
}
}
```

Compiler Error: □

OR

Write Output:

```
public class X {
public int orange = 0;
public void bar() {
orange++;
System.out.print( "B" );
}
}
public class Y {
public void banana() {
X x = new X();
System.out.print( "C" );
x.orange;
}
public static void main( String []
args ) {
Y y = new Y();
System.out.print( "D" );
y.banana();
}
}
```

Compiler Error: □

OR

Write Output:

b) [**20 marks**] What do the following code fragments print? In some cases, spaces and empty lines are important in the final answer and must be clearly shown. To make this clear, you can indicate the presence of a space using the underscore character ('_') and indicate an empty line with the text “<Blank line>”. If the code generates an error during execution, indicate/write the nature of the error. However, you must still note any output the program makes before the error occurs. Note, the symbol: ↵ refers to the newline character when it appears in the input.

```
public class X {
public static void main( String []
args ) {
System.out.println(“Hi + \”SE3001\” +
Students”);
}
}
```

Write Output:

```
public class X {
public static void main( String []
args ) {
System.out.println(“Hi ”);
System.out.print(“CS133”);
System.out.println(“Students”);
}
}
```

Write Output:

```
public class X {
public static void main( String []
args ) {
String s="Hi SE3001 Students!";
String t=null;
System.out.println(s.substring(3,17));
s=t;
System.out.println(s.charAt(1));
}
}
```

Write Output:

```
public class Rectangle {
public static int area(int height, int
width) {
return height * width;
}
public static void main( String []
args ) {
System.out.println(“The result of area
is: ” +
Rectangle.area(4, 5) );
}
}
```

Write Output:

```
public class X {
public static void main( String []
args ) {
int mark=85;
if(mark > 90) {
System.out.println("Excellent");
}else if(mark < 90 || mark > 80) {
System.out.println("Good job!");
}else if (mark == 85 ) {
System.out.println("Good job! your
mark is
exactly 10 marks above the average");
}else if (mark > 70) {
System.out.println("Doing okay.");
}else if(mark > 50) {
System.out.println("Passed");
} else {
System.out.println("Failed");
}
}
}
```

Write Output:

```
public class X {
public static void main( String []
args ) {
for (int i=10; i >0 ; i=i/2) {
System.out.print(i + “,”);
}
System.out.println(“Good job!”);
}
}
```

Write Output:

```
import java.util.*;
public class X {
public static void main( String []
args ) {
String name;
int first;
String second;
Scanner keyBoardInput=new
Scanner(System.in);
System.out.println("Enter your
name:");
name=keyBoardInput.nextLine();
System.out.println("Enter two numbers
on different lines:");
first=keyBoardInput.nextInt();
second=keyBoardInput.nextLine();
System.out.println("Hi:" + name);
System.out.println("first + second = "
+ first + second);
}
}
```

Input:

```
Anne John ↵
13 ↵
31 ↵
```

Write Output:

```
public class C {
public int x;
}
public class D {
public static void f (C c, int y) {
System.out.println(c.x);
c.x = y;
y++;
System.out.println(c.x);
c = new C();
c.x = y+2;
System.out.println(c.x);
}

public static void main (String[]
args) {
int z = 4;
C c = new C();
c.x = 3;
System.out.println(c.x);
f(c, z);
System.out.println(c.x);
System.out.println(z);
}
}
```

Write Output:

```
public class SimpleCalc {
public int value;
public void calculate( ) { value +=
7; }
 }

public class MultiCalc extends
SimpleCalc {
public void calculate( ) { value +=
3; }
public void calculate( int multipier)
{
calculate( );
super.calculate( );
value *= multipier;
}
public static void main(String[] args)
{
MultiCalc calculator = new
MultiCalc( );
calculator.calculate(2);
System.out.println(“ Value is: “ +
calculator.value);
}}
```

Write Output:

```
class Pizza {
java.util.ArrayList toppings;
public final void addTopping(String
topping) {
toppings.add(topping);
}
}
public class PepperoniPizza extends
Pizza {
public void addTopping(String topping)
{
System.out.println("Cannot and
Uoppings");
}
public static void main(String[] args)
{
Pizza pizza = new PepperoniPizza();
Pizza.addTopping("Mushrooms");
}
}
```

Write Output:

Question 2: [CLO1] [6 marks]

In this problem you are asked to write a simple class to represent elevators. An elevator has a current floor, a number of floors, a current number of passengers, and a maximum capacity of passengers.

An elevator:

- is constructed by specifying:
    - the total number of floors in the building
    - the maximum elevator capacity
    - that the elevator initially doesn’t have any passengers,
    - and that the elevator is initially located on the bottom floor
- can move one floor up if not on the top floor
- can move one floor down if it is not on the bottom floor
- can accept a certain number of passengers (up to its maximum capacity)
- can drop off a certain number of passengers (no more than it actually has)
- can tell us which floor it's on.

Question 3: [CLO1] [5 marks]

Write a program that input English words using JOptionPane dialog and store the inputted words to the suitable collection. The program should have two methods: one to add the word into the suitable collection and other method will display all the stored words from the collection in the ascending order in an alert box (**Note:** *you cannot use Collections.sort() method for this program and your suitable collection should store distinct words only*)

Question 4: [CLO1] [5 marks]

Create a generic class that accept only numbers including short, integers, floats, and doubles only and must contains two generic methods namely **oddSum** and **evenSum** that returns the sum of odd numbers and even numbers respectively that stored within an arraylist of accepted numbers.