Advanced Programming (CSE201), Midsem Quiz
Time allocated: 03:00pm – 4:00pm (1 hour), Total Marks: 20
(No Submissions are allowed beyond 4.10pm)

Instructions:

- This is a closed book quiz.
- Only reasonable and clearly mentioned assumptions (if any) would be accepted.
- For justifications, please be as concise as possible (2-3 sentences only)
- **IIIT plagiarism policy is applicable if any such cases found**
- **Write your answers on a plain sheet that you can upload by taking a picture of the same (ensure low resolution so that the upload size is smaller)**
- **You can either submit your solutions on the below Google Form link https://docs.google.com/forms/d/e/1FAIpQLSe6yba9jd_k5KnHt071_9fTYj8VJWAnSgyHcxHUE21N2aAPIw/viewform?usp=sf_link or email the quiz solutions to "ap-m2020-submission@iiitd.ac.in" or upload on google classroom . Only one mode of submission!**
- **Subject of the mail should be the Midsem Quiz.**
- **We will not consider any submission that is submitted beyond 4.10pm. It is your responsibility to ensure you email it or submit through google form or through google classroom on time. Please ensure you have proper internet connectivity as we are giving you sufficient extra time to send the email or upload your solutions either on google classroom or on google form.**

---

1) The general intent of cloning is to satisfy the following three soft requirements.

- x.clone() != x
- x.clone().equals(x)
- x.clone().getClass() == x.getClass()

a) In the following code fragment, indicate the lines that satisfy each of the above three requirements. In addition to that, also explain why/how they satisfy the three requirements? **[6 marks]**

```
1. public Point3D clone() {
2.     Point3D p = (Point3D) super.clone();
3.     p.z = this.z;
4.      return p;
5. }
```

Answer:
Point-1 corresponds to line 2 [+1 marks]
Explanation: A new instance/reference has been created [+1 marks]
Point-2 corresponds to lines 2, 3 [0.5+0.5 marks]
Explanation: Initialization of all the instance variables with the values of the object to be cloned [+1 marks]
Point-3 corresponds to line 2 [+1 marks]
Explanation: getClass() would return the actual type of the object that gets created and line-2 returns the actual type. [+1 marks]

b) Does the below clone() method violate any of the three requirements mentioned above? Explain your answer. **[2 marks]**

```
1. public class Organization implements Cloneable {

2.      private String orgName;
3.      private List<String> employeeNames;

4.      @Override
5.      public Organization clone() {
6.              try {
7.                      Organization copy = (Organization) super.clone();
8.                      return copy;
9.              } catch(CloneNotSupportedException e) {
10.                     e.printStackTrace();
11.                     return null;
12.             }
13.     }
14. }
```

Answer: This code will not do a deep copy of the list. [+1 marks]
So the first point from the soft requirements (not sharing the reference) would be violated. [+1 marks]

Q2) Consider the code given below. **[4 marks]**

a) Write the output of the two print statements.
b) What is the value of the instance variables after deserialization? Explain your answer.
c) In case this code results in an error, explain the reason.

```
1. public class Organization implements Serializable {

2.      private static final long serialVersionUID = 20L;
3.      private int orgID;
4.      private transient List<String> employeeNames;
5.      private transient double profit;


6.      public Organization(int orgID, List<String> employeeNames, double profit) {
7.              this.orgID = orgID;
8.              this.employeeNames = employeeNames;
9.              this.profit = profit;
10.     }

11.     public void serialize() throws FileNotFoundException, IOException {
12.             ObjectOutputStream out = null;
13.             try {
14.                     out = new ObjectOutputStream(new FileOutputStream("out.txt"));
15.                     out.writeObject(this);
16.             } finally {
17.                     out.close();
18.             }
19.     }

20.     public Organization deserialize() throws FileNotFoundException, IOException, ClassNotFoundException {
21.             ObjectInputStream in = null;
22.             Organization org = null;
23.             try {
24.                     in =  new ObjectInputStream (new FileInputStream("out.txt"));
25.                     org = (Organization) in.readObject();
26.             } finally {
27.                     in.close();
28.             }
29.             return org;
30.     }

31.     @Override
32.     public String toString() {
33.             return orgID + " " + employeeNames.toString() + " " + profit;
34.     }

35. }
```

```
1.  public static void main(String[] args) {
2.                  List<String> empList = new ArrayList<String>();
3.                  empList.add("John");
4.                  empList.add("Mike");
5.                  Organization org = new Organization(10, empList, 100.02);
6.                  System.out.println(org);
7.                  try {
8.                          org.serialize();
9.                          Organization deserializedOrg = org.deserialize();
10.                         System.out.println(deserializedOrg);
11.                 } catch (Exception e) {
12.                         e.printStackTrace();
13.                 }
14.     }
```

Answer:

[No Partial Marking]

a) **Line-6 output: 10 [John, Mike] 100.02 [+1 marks]**
   Line-10 output: Results in a NullPointerException [+1 marks]
b) **orgID is 10, employeeNames is null and profit is 0.0. The last two get the default values (null and 0.0 respectively) because their state has not been saved (they are transient) and while deserializing, the default constructor assigns the default values to both the instance variables. [+1 marks]**
c) **Yes, this code results in a NullPointerException [+0.5 marks] because employeeNames is null and in toString(), the call employeeNames.toString() results in a NullPointerException. [+0.5 marks]**

Q3) Explain why Java does not support multiple inheritance by taking the reference of the diamond problem. Feel free to use the resources on the Internet for this question, but include appropriate citation(s) in your answer. You must write your answer in your own words instead of copy-pasting from the Internet. **[2 marks]**

Answer: In short, you need to write that two (or more) parents of a class can inherit from a common parent [+1 marks] and in this case, there would be confusion as to which class a method belongs to. [+1 marks]

Any explanation with a diagram would work.

All the following links explain this problem with the help of an example.

- https://en.wikipedia.org/wiki/Multiple_inheritance
- https://www.geeksforgeeks.org/java-and-multiple-inheritance/
- https://www.javatpoint.com/what-is-diamond-problem-in-java
- https://www.tutorialspoint.com/what-is-diamond-problem-in-case-of-multiple-inheritance-in-java

Q4) Rewrite the following class so that it becomes immutable. Do not assign values to the instance variables directly at the time of declaration. **[2 marks]**

```
1.  public class MakeThisImmutable {

2.          public int id;
3.          protected String name;
4.          public Employee emp;

5.          public int getID() {
6.                  return id;
7.          }
8.          public void setID(int id) {
9.                  this.id = id;
10.         }
11.         public String getName() {
12.                 return name;
13.         }
14.         public void setName(String name) {
15.                 this.name = name;
16.         }
17.         public Employee getEmployee() {
18.                 return new Employee();
19.         }
20.         public void setEmployee(Employee emp) {
21.                 this.emp = emp;
22.         }
23. }
```

```
1. public class Employee {
2.          private String name;

3.          public Employee() {
4.                  name = "noname";
5.          }
6.          public String getName() {
7.                  return name;
8.          }
9.          public void setName(String name) {
10.                 this.name = name;
11.         }
12. }
```

Answer:

```
public final class MakeThisImmutable {

        private final int id;
        private final String name;
        private final Employee emp;

        public MakeThisImmutable() {
                id = 123;
                name = "John";
                emp = new Employee();
        }
        public int getID() {
                return id;
        }
        public String getName() {
                return name;
        }
        public Employee getEmployee() {
                return new Employee();
        }
}
```

a) **Encapsulation [+0.5 marks]**
All class attributes are marked as private

b) **Final [+0.5 marks]**
All class attributes are marked as Final

c) **There should not be any Setters [+0.5 marks]**

d) **Class should be made final [+0.5 marks]**

Q5) Construct the skeleton code that would lead to the following stacktrace. We expect the exact class and method names. Include the line numbers wherever appropriate. **Do not use**

loops. Handle the exceptions, i.e., there should be try and catch blocks. Using your skeleton code, explain how it would end up with each of the three exceptions listed here. **[4 marks]**

```
1. java.lang.NullPointerException
2.         at University.getCourses(University.java:27)
3.         at University.enrollIntoCourse(University.java:17)
4.         at University.teachCourse(University.java:9)
5.         at University.main(University.java:37)
6.  java.lang.NullPointerException
7.         at University.getCourses(University.java:27)
8.         at University.enrollIntoCourse(University.java:17)
9.         at University.teachCourse(University.java:9)
10.        at University.main(University.java:37)
11.  java.lang.NullPointerException
12.        at University.getCourses(University.java:27)
13.        at University.enrollIntoCourse(University.java:17)
14.        at University.teachCourse(University.java:9)
15.        at University.main(University.java:37)
```

```
public class University {

        public void teachCourse() {
                try {
                        enrollIntoCourse();
                } catch (NullPointerException e) {
                        e.printStackTrace();
                }
        }

        public void enrollIntoCourse() {
                try {
                        getCourses();
                } catch (NullPointerException e) {
                        e.printStackTrace();
                        throw e;
                }
        }

        public List<String> getCourses() {
                String c = null;
                try {
                        c.length();
                } catch (NullPointerException e) {
                        e.printStackTrace();
                        throw e;
                }
                return null;
        }

        public static void main(String[] args) {
                University u = new University();
                u.teachCourse();
        }
}
```

a) **The exact sequence of method calls as given in the stacktrace [+1 marks]**
b) **Use of try-catch block and the catch should have NullPointerException [+1 marks]**
c) **Use of printStackTrack() [+0.5 marks]**
d) **Exception should be rethrown twice [+0.5 marks]**
e) **Line numbers for the methods [+1 marks]**