a) Your colleague has just started to learn Java Binary Input / Output and tried to implement a console application that read/write primitive and String data types only from/to a file in a Windows operating system. Currently, your colleague needs your help to complete the implementation where it is commented with TODO.

The following Table 1 is a Java class called ChildrenDetails.java which your colleague is currently implementing.

ChildrenDetails.java

```
1.  public class ChildrenDetails {
2.    public static void main(String[] args)
3.        throws FileNotFoundException, IOException {
4.      Scanner scanner = new Scanner(System.in);
5.      // The directory is valid and JaiChildren.dat already exists
6.      File file = new File("JaiChildren.dat");
7.      try(
8.        // TODO 1
9.      ) {
10.       System.out.println("Children Details in the file.");
11.       while(readData.available() != 0) {
12.         System.out.println("######################");
13.         System.out.println("Name: " + readData.readUTF());
14.         System.out.println("Year of Birth: " + readData.readInt());
15.         System.out.println("Month of Birth: " + readData.readInt());
16.         System.out.println("Day of Birth: " + readData.readInt());
17.         System.out.println("######################");
18.       }
19.     }
20.     try(
21.       // TODO 2
22.     ) {
23.       System.out.println("Filling in more children details:");
24.       System.out.println("Name:");
25.       String name = scanner.nextLine();
26.       System.out.println("Date of Birth (dd/mm/yyyy):");
27.       String[] dob = scanner.nextLine().split("/");
28.       int day = Integer.parseInt(dob[0]);
29.       int month = Integer.parseInt(dob[1]);
30.       int year = Integer.parseInt(dob[2]);
31.       // TODO 3
32.     }
33.     scanner.close();
34.   }
35. }
```

Table 1

i. In Table 1, implement the appropriate InputStream object at Line 8 commented with TODO 1 that will read the File object, file, in Line 6. **[3 marks]**

ii. In Table 1, implement the appropriate OutputStream object at Line 21 commented with TODO 2 that will append the File object, file, in Line 6. **[3 marks]**

iii. In Table 1, implement the appropriate statements at Line 31 commented with TODO 3 that will write data in Line 25, Line 28, Line 29, and Line 30 once using the OutputStream object created in **Question a)ii.** to the File object, file, in Line 6 accordingly. The data written into the file must follow the order of it being read in Line 13 – Line 16 and its integrity still maintained. **[4 marks]**

b) The following Table 2, Table 3 and Table 4 are **three (3)** Java classes called StudentServer.java, StudentClient.java and Student.java respectively.

StudentServer.java

```
1.   public class StudentServer {
2.     public static void main(String[] args) throws Exception {
3.
4.     }
5.   }
```

Table 2

StudentClient.java

```
1.   public class StudentClient {
2.     public static void main(String[] args) throws Exception {
3.
4.     }
5.   }
```

Table 3

Student.java

```
1.   public class Student implements Serializable {
2.     String id;
3.     String name;
4.     int intake;
5.
6.     public Student(String id, String name, int intake) {
7.       this.id = id;
8.       this.name = name;
9.       this.intake = intake;
10.    }
11.
12.    public String toString() {
13.      return "(" + id + ") " + name + " - Intake " + intake;
14.    }
15.  }
```

Table 4

i. Table 2 is a server application. Complete the implementation of Table 2 based on the following:

When the server application starts, it will listen to port number 9521. The server application should accept connection from a client application, and it should be able to send Java objects to and receive Java objects from the client application. (Note: You are not required to rewrite the whole class, just write the code statements in proper order. You are not required to handle any exceptions being thrown by any of the statements or import any of the classes.)

[4 marks]

ii. Table 3 is a client application that connects to the server application (Table 2). Complete the implementation of Table 3 based on the following:
It will request a connection to the server application hosted in IP address 212.200.1.30. The client application should be able to send Java objects to and receive Java objects from the server application. (Note: You are not required to rewrite the whole class, just write the code statements in proper order. You are not required to handle any exceptions being thrown by any of the statements or import any of the classes.)

[3 marks]

iii. Implement the communications between the server application (Table 2) and the client application (Table 3). Add further implementations to Table 2 and Table 3 based on the following order (Note: State the implementation is for server application and client application respectively):

1. The server application sends Student, Table 4, object with the data (id: "20FTT9991", name: "Abu", intake: 9) to the client application.
2. The client application modifies the received Student object, instance variable, intake, to value 10.
3. The client application sends the Student object back to the server application.

[5 marks]

c) The following Table 5 is a Java class called MainController.java.

MainController.java
1. @Controller 2. public class MainController { 3. 4. }

Table 5

i. Implement a method in Table 5 to handle Uniform Resource Identifier (URI) "/" where it will respond with a literal String "<h1>Welcome</h1>". (Note: You are not required to rewrite the whole class.) **[4 marks]**

ii. Implement a method in Table 5 to handle Uniform Resource Identifier (URI) "/{message}" where it will respond with a literal String of the path variable. (Note: You are not required to rewrite the whole class.) **[5 marks]**

d) The following Table 6 and Table 7 are **two (2)** Java classes called Employee.java and HomeController.java respectively and Figure 1 is a command line (terminal) screenshot of a table details queried in MariaDB Database Management System server that was generated by Table 6 included in a Spring Web Application with Hibernate.

Employee.java

```
1.  public class Employee {
2.
3.  }
```

Table 6

HomeController.java

```
1.  @Controller
2.  public class HomeController {
3.    @Autowired
4.    EmployeeRepository employeeRepository;
5.  }
```

Table 7

```
MariaDB [jailani_nep]> describe employees;
+-----------+--------------+------+-----+---------+-------+
| Field     | Type         | Null | Key | Default | Extra |
+-----------+--------------+------+-----+---------+-------+
| id        | int(11)      | NO   | PRI | NULL    |       |
| full_time | bit(1)       | NO   |     | NULL    |       |
| name      | varchar(255) | YES  |     | NULL    |       |
| salary    | double       | NO   |     | NULL    |       |
+-----------+--------------+------+-----+---------+-------+
4 rows in set (0.002 sec)
```

Figure 1

i. Complete Table 6 implementation by making the class a Database entity that will be used by Hibernate to create the Database table (Figure 1) with its field **and** constraints accordingly. **[10 marks]**

ii. Implement the Data Access Object interface that will be use by Table 7, Line 4 to perform generic create, read, update **and** delete (CRUD) operation to the database table, employees. **[4 marks]**

e) Your colleague has just started to learn Java Binary Input / Output and tried to implement a console application that read/write object data types from/to a file in a Windows operating system. Currently, your colleague needs your help to complete the implementation where it is commented with TODO. The following Table 1 and Table 2 are **two (2)** Java classes called ChildrenDetails.java and Children.java respectively which your colleague is currently implementing.

ChildrenDetails.java

```
1.  public class ChildrenDetails {
2.    public static void main(String[] args) throws
3.        FileNotFoundException, IOException, ClassNotFoundException {
4.      Scanner scanner = new Scanner(System.in);
5.      // The directory is valid and JaiChildren.dat already exists
6.      File file = new File("JaiChildren.dat");
7.      try(
8.        // TODO 1
9.      ) {
10.       System.out.println("Children Details in the file.");
11.       while(readData.available() != 0) {
12.         System.out.println((Children) readData.readObject());
13.       }
14.       System.out.println("######################");
15.     }
16.     try(
17.       // TODO 2
18.     ) {
19.       System.out.println("Filling in more children details:");
20.       System.out.println("Name:");
21.       String name = scanner.nextLine();
22.       System.out.println("Date of Birth (dd/mm/yyyy):");
23.       String[] dob = scanner.nextLine().split("/");
24.       int day = Integer.parseInt(dob[0]);
25.       int month = Integer.parseInt(dob[1]);
26.       int year = Integer.parseInt(dob[2]);
27.       // TODO 3
28.     }
29.     scanner.close();
30.   }
31. }
```

Table 1

Children.java

```
1.  public class Children implements Serializable {
2.    String name;
3.    int yearOfBirth;
4.    int monthOfBirth;
5.    int dayOfBirth;
6.
7.    public Children(String name, int yearOfBirth,
8.        int monthOfBirth, int dayOfBirth) {
9.      this.name = name;
10.     this.yearOfBirth = yearOfBirth;
11.     this.monthOfBirth = monthOfBirth;
12.     this.dayOfBirth = dayOfBirth;
13.   }
14.
15.   public String toString() {
16.     return name + " -> " + dayOfBirth + "/" +
17.         monthOfBirth + "/" + yearOfBirth;
18.   }
19. }
```

Table 2

i. In Table 1, implement the appropriate InputStream object at Line 8 commented with TODO 1 that will read the File object, file, in Line 6. **[3 marks]**

ii. In Table 1, implement the appropriate OutputStream object at Line 17 commented with TODO 2 that will append the File object, file, in Line 6. **[3 marks]**

iii. In Table 1, implement the appropriate statements at Line 27 commented with TODO 3 that will write a Children, Table 2, object once with the data in Line 21, Line 24, Line 25, Line 26 using the OutputStream object created in **Question e)ii**. to the File object, file, in Line 7 accordingly. The data written into the file must follow the order of it being read in Line 13 **and** its integrity still maintained. **[3 marks]**

iv. There are **three (3)** Children objects already added to Children.dat (Table 1, Line 6) in the following order and data:

a. First Children object:
   name: "Abu", yearOfBirth: 2015, monthOfBirth: 10, dayOfBirth: 22

b. Second Children object:
   name: "Bakar", yearOfBirth: 2017, monthOfBirth: 5, dayOfBirth: 2

c. Third Children object:
   name: "Curi", yearOfBirth: 2019, monthOfBirth: 1, dayOfBirth: 15

Produce the output generated by the code in Table 1, Line 11 – Line 15. **[3 marks]**

f) The following Table 2 and Table 3 are **two (2)** Java classes called StudentServer.java and StudentClient.java respectively.

StudentServer.java

```
1.   public class StudentServer {
2.     public static void main(String[] args) throws Exception {
3.
4.     }
5.   }
```

Table 2

StudentClient.java

```
1.   public class StudentClient {
2.     public static void main(String[] args) throws Exception {
3.
4.     }
5.   }
```

Table 3

Student.java

```
1.   public class Student implements Serializable {
2.     String id;
3.     String name;
4.     int intake;
5.
6.     public Student(String id, String name, int intake) {
7.       this.id = id;
8.       this.name = name;
9.       this.intake = intake;
10.    }
11.
12.    public String toString() {
13.      return "(" + id + ") " + name + " - Intake " + intake;
14.    }
15.  }
```

Table 4

i. Table 2 is a server application. Complete the implementation of Table 2 based on the following:

When the server application starts, it will listen to port number 9991. The server application should accept connection from a client application, and it should be able to send to and receive Java primitive and String data types only from the client application. (Note: You are not required to rewrite the whole class, just write the code statements in proper order. You are not required to handle any exceptions being thrown by any of the statements or import any of the classes.)

[4 marks]

ii. Table 3 is a client application that connects to the server application (Table 2). Complete the implementation of Table 3 based on the following:
It will request a connection to the server application hosted in IP address 107.212.10.203. The client application should be able to send to and receive Java primitive and String data types only from the server application. (Note: You are not required to rewrite the whole class, just write the code statements in proper order. You are not required to handle any exceptions being thrown by any of the statements or import any of the classes.)

[3 marks]

iii. Implement the communications between the server application (Table 2) and the client application (Table 3). Add further implementations to Table 2 and Table 3 based on the following order (Note: State the implementation is for server application and client application respectively):

4. The client application sends String data, "20FTT9991" to the server application.
5. The server application sends String data, "Abu" and integer data, 10 to the client application. **[5 marks]**

g) The following Table 5 is a Java class called MainController.java.

MainController.java

```
1.  @Controller
2.  public class MainController {
3.
4.  }
```

Table 5

i. Implement a method in Table 5 to handle Uniform Resource Identifier (URI) "/home" where it will respond with a literal String "<h1>Welcome</h1><h2>Hello</h2>". (Note: You are not required to rewrite the whole class.)

[4 marks]

ii. Implement a method in Table 5 to handle Uniform Resource Identifier (URI) "/message" with Hypertext Transfer Protocol (HTTP) Get Request parameters name and message where it will respond with a literal String of the concatenation of HTTP Get Request parameters name and message. (Note: You are not required to rewrite the whole class.)

[7 marks]

h) The following Table 6 and Table 7 are Java classes called HomeController.java and Stock.java respectively and Table 8 is a Hypertext Markup Language (HTML) called stocks.html, that are used for Spring Web Application (with Thymeleaf).

HomeController.java

```
1.  @Controller
2.  public class HomeController {
3.    List<Stock> stocks = (List<Stock>) Arrays.asList(
4.      new Stock("Laptop", 10, 1999.99),
5.      new Stock("Solid State Drive", 100, 99.99),
6.      new Stock("SD Card", 200, 30.99),
7.      new Stock("Monitor", 100, 299.99)
8.    );
9.  }
```

Table 6

Stock.java

```
1.   public class Stock {
2.     private String name;
3.     private int quantity;
4.     private double price;
5.
6.     public Stock(String name, int quantity, double price) {
7.       this.name = name;
8.       this.quantity = quantity;
9.       this.price = price;
10.    }
11.
12.    public String getName() { return name; }
13.    public void setName(String name) { this.name = name; }
14.    public int getQuantity() { return quantity; }
15.    public void setQuantity(int quantity) { this.quantity = quantity; }
16.    public double getPrice() { return price; }
17.    public void setPrice(double price) { this.price = price; }
18.  }
```

Table 7

stocks.html

```
1.  <!DOCTYPE html>
2.  <html>
3.  <head>
4.  <title>NS4307 NEP</title>
5.  </head>
6.  <body>
7.  <h1>List of Stocks</h1>
8.  <table>
9.  <tr>
10. <th>Name</th>
11. <th>Quantity</th>
12. <th>Price</th>
13. </tr>
14. <tr>
15. <td></td>
16. <td></td>
17. <td></td>
18. </tr>
19. </table>
20. </body>
21. </html>
```

Table 8

i. Implement a method in Table 6 to handle Uniform Resource Identifier (URI) "/stocks" where it will response with the HTML file, Table 8 with data from Table 6, Line 3 – Line 8 passed into the HTML file. (Note: You are not required to rewrite the whole class.)

[5 marks]

ii. Complete the implementation of Line 14 to Line 18 of the HTML file, Table 8, which it should display all the data passed into it, Table 6, Line 3 – Line 8. Each Table 7 object should be displayed as table row and each field of Table 7 object is displayed in its respective table column. (Note: You are not required to rewrite the whole html file.)

[6 marks]