ڤوليتيكنيك بروني
POLITEKNIK BRUNEI

END-OF-SEMESTER EXAMINATION (ALTERNATE)

SEMESTER 2, 2022-23

SCHOOL OF INFORMATION & COMMUNICATION TECHNOLOGY

NS4307

NETWORK PROGRAMMING

TIME ALLOWED: 2 HOURS

INSTRUCTIONS TO CANDIDATES:

- **DO NOT TURN OVER THE PAGE UNTIL YOU ARE TOLD TO DO SO.**
- Write down the details required on the front cover of the answer booklet provided. Make sure you complete the details on any additional answer booklets that you use.
- Answer ALL questions.
- The total mark for this paper is **70**. The number of marks for each question or part question is shown in brackets.
- Your answer booklets and any other paper used in the examination must be handed in to the invigilators before leaving the room or hall. You may take your own examination paper with you unless otherwise instructed.
- Write in **dark blue** or **black** ink pen.

MATERIALS SUPPLIED:

Answer booklet (1).

MATERIALS PERMITTED:

Calculator.

STRUCTURED QUESTIONS [TOTAL MARKS: 70]

Answer **ALL** questions.

QUESTION 1 [Total marks: 14]

a) "Version control is a system that records changes to a file or set of files over time." Explain why that is the case. **[1 mark]**

b) There are **three (3)** types of Version Control Systems (VCS) and one of them is Local Version Control Systems. List the other **two (2)** types of VCS. **[1 mark]**

c) Illustrate the basic structure of Local Version Control Systems. **[3 marks]**

d) Explain Git's **three (3)** main states: modified, staged **and** committed, that your files can reside in. **[3 marks]**

e) State any **three (3)** typical features in Version Control Systems. **[3 marks]**

f) List **and** explain **two (2)** ways of using Git to update the development environment. **[3 marks]**

QUESTION 2 [Total marks: 14]

a) "Java Binary Input/Output is more efficient than Text Input/Output." Justify the statement. **[1 mark]**

b) The following Table 1 and Table 2 are two Java classes named SaveStudentFile.java and LoadStudentFile.java, respectively, with an empty binary file, Table 3, named student.dat.

SaveStudentFile.java

```java
public class SaveStudentFile {
  public static void main(String[] args) {
    try(DataOutputStream output = new DataOutputStream(new
        FileOutputStream("student.dat", false))) {
      output.writeInt(123);
      output.writeUTF("Jailani");
      output.writeDouble(98.5);
      output.writeUTF("SICT");
    } catch (IOException e) {
      e.printStackTrace();
    }
  }
}
```

Table 1

LoadStudentFile.java

```java
public class LoadStudentFile {
  public static void main(String[] args) {
    try(/*TODO1*/) {
    /*TODO2*/
  } catch (IOException e) {
    e.printStackTrace();
  }
  }
}
```

Table 2

student.dat

Table 3

~~i.~~ Complete the implementation of LoadStudentFile.java (Table 2) by replacing /*TODO1*/ and /*TODO2*/ with proper input streams to read and print to console all the contents of student.dat (Table 3) after SaveStudentFile.java (Table 1) was executed once. (Note: you are not required to rewrite the whole class, just write the codes needed in /*TODO1*/ and /*TODO2*/). **[4 marks]**

[Turn over

c) 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 4 and Table 5 are **two (2)** Java classes called ChildrenDetails.java and Children.java, respectively, which your colleague is currently implementing.

ChildrenDetails.java

```
public class ChildrenDetails {
  public static void main(String[] args) throws
      FileNotFoundException, IOException, ClassNotFoundException {
    Scanner scanner = new Scanner(System.in);
    // The directory is valid and JaiChildren.dat already exists
    File file = new File("JaiChildren.dat");
    try(
      // TODO 1
    ) {
      System.out.println("Children Details in the file.");
      while(readData.available() != 0) {
        System.out.println((Children) readData.readObject());
      }
      System.out.println("######################");
    }
    try(
      // TODO 2
    ) {
      System.out.println("Filling in more children details:");
      System.out.println("Name:");
      String name = scanner.nextLine();
      System.out.println("Date of Birth (dd/mm/yyyy):");
      String[] dob = scanner.nextLine().split("/");
      int day = Integer.parseInt(dob[0]);
      int month = Integer.parseInt(dob[1]);
      int year = Integer.parseInt(dob[2]);
      // TODO 3
    }
    scanner.close();
  }
}
```

Table 4

Children.java

```java
public class Children implements Serializable {
  String name;
  int yearOfBirth;
  int monthOfBirth;
  int dayOfBirth;

  public Children(String name, int yearOfBirth,
      int monthOfBirth, int dayOfBirth) {
    this.name = name;
    this.yearOfBirth = yearOfBirth;
    this.monthOfBirth = monthOfBirth;
    this.dayOfBirth = dayOfBirth;
  }

  public String toString() {
    return name + " -> " + dayOfBirth + "/" +
        monthOfBirth + "/" + yearOfBirth;
  }
}
```

Table 5

i. Based on Table 4, implement the appropriate InputStream object in Line 8, commented with TODO 1, that will read the File object, file, in Line 6.

[3 marks]

ii. Based on Table 4, implement the appropriate OutputStream object in Line 17, commented with TODO 2, that will append the File object, file, in Line 6.

[3 marks]

iii. Based on Table 4, implement the appropriate statements in Line 27 commented with TODO 3 that will write Children.java, Table 5, object once with the data in Line 21, Line 24, Line 25, and Line 26, using the OutputStream object created in **Question 2.c)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 **with** its integrity still maintained.

[3 marks]

[Turn over

QUESTION 3 [Total marks: 14]

a) Explain Threads in Java. **[2 marks]**

b) The following Table 6 and Table 7 are **two (2)** Java classes called StudentServer.java and StudentClient.java, respectively.

`StudentServer.java`

```java
public class StudentServer {
  public static void main(String[] args) throws Exception {

  }
}
```

Table 6

`StudentClient.java`

```java
public class StudentClient {
  public static void main(String[] args) throws Exception {

  }
}
```

Table 7

i. Table 6 is a server application. Complete the implementation of Table 6 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: write only the code statements in proper order. You are not required to rewrite the whole class, handle any exceptions being thrown by any of the statements, or import any of the classes.) **[4 marks]**

ii. Table 7 is a client application that connects to the server application (Table 6). Complete the implementation of Table 7 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: write only the code statements in proper order. You are not required to rewrite the whole class, 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 6) and the client application (Table 7). Add further implementations to Table 6 and Table 7 based on the following order (Note: State whether the implementation is for server application or client application):

1. The client application sends String data, “20FTT9991”, to the server application.
2. The server application sends String data, “Abu”, and integer data, 10, to the client application. **[5 marks]**

QUESTION 4 [Total marks: 14]

a) Define Thymeleaf. **[1 mark]**

b) The following Table 8 is a HyperText Markup Language (HTML) file called index.html and Table 9 is a Java class called ControllerClass that are used for Spring Application (with Thymeleaf).

index.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head></head>
<body>
<h1 th:text="${u.id} + ' ' + ${u.name}"></h1>
<p th:text="${u.percentage}"></p>
<p th:text="${u.active}"></p>
</body>
</html>
```

Table 8

ControllerClass.java

```
@Controller
public class ControllerClass {
  @RequestMapping("/")
  public String home(ModelMap modelMap) {
    User user = new User(551, "Abu", 77.5, true);
    modelMap.put("u", user);
    return "index";
  }
}
```

Table 9

[Turn over

By analysing the above codes, create a Plain Old Java Object (POJO) class with only instance variables that are stated including constructor, getter and setter methods.

[13 marks]

QUESTION 5 [Total marks: 14]

a) Before a Spring Web Application with Hibernate can connect to a database management system, the web application needs to be configured first. One of the properties that can be configured is spring.jpa.hibernate.ddl-auto. Explain the **four (4)** values: create, create-drop, update and validate, that can be assigned to the property.

[4 marks]

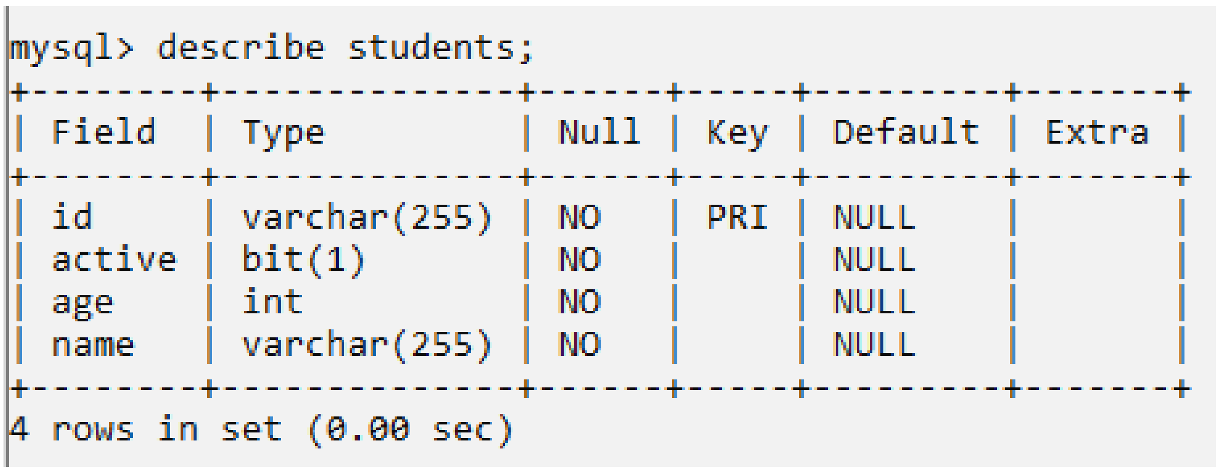
b) The following Table 10 is a Java class called Student.java and Table 11 is a command line (terminal) screenshot of a table details, queried in MySQL Database Management System server that are used for Spring Web Application with Hibernate.

Student.java

```
public class Student {

}
```

Table 10



```
mysql> describe students;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| id     | varchar(255) | NO   | PRI | NULL    |       |
| active | bit(1)       | NO   |     | NULL    |       |
| age    | int          | NO   |     | NULL    |       |
| name   | varchar(255) | NO   |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
4 rows in set (0.00 sec)
```

Table 11

Rewrite the Plain Old Java Object class, Student.java (Table 10), by making the class a Database entity that will be used by Hibernate to create the Database table (Table 11), with its field and constraints accordingly.

[10 marks]

[END OF PAPER]