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

(MARKING SCHEME)

PREPARED BY MOHAMMAD JAILANI BIN HAJI ABDUL RAHMAN

INSTRUCTIONS TO MARKERS:

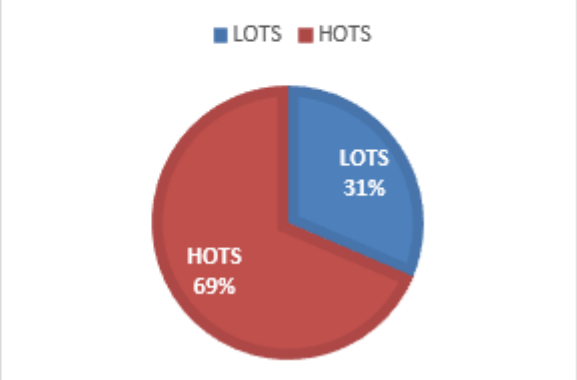
- Students have to answer ALL questions
- Students have 2 Hours to attempt this paper unless otherwise specified.
- The total mark for this paper is **70**. The number of marks for each question or part question is shown in brackets.
- The answers to the paper are shown in blue colour fonts.
- Keywords/ key points are shown in bold/underlined.
- Answer in italic represents an alternate answer.
- Only answers written in blue or black ink will be assessed except for diagram, where students are allowed to use pencil.
- The Blooms indicators after each question represents the Bloom's taxonomy classifying the cognitive level required in answering the questions.
- The assessment specification table provides the summary of the paper.
- Only use **RED** colour ink to mark.

No	Question	Marks	Bloom's Taxanomy	Source	Source Q
1	1.a.	1	Remember	Resit	1.a.
2	1.b.	1	Remember	Actual	1.b.
3	1.c.	3	Apply	Actual	1.c.
4	1.d.	3	Remember	ctual	1.d.
5	1.e.	3	Remember	esit	1.d.
6	1.f.	3	Remember	Resit	1.f.
7	2.a.	1	Remember	Actual	2.a.
8	2.b.	4	Evaluate	Actual	2.b.
9	2.c.i.	3	Evaluate	Actual	2.c.i.
10	2.c.ii.	3	Evaluate	Actual	2.c.ii.
11	2.c.iii.	3	Evaluate	Actual	2.c.iii.
12	3.a.	2	Remember	Resit	3.a.
13	3.b.i.	4	Evaluate	Resit	3.b.i.
14	3.b.ii.	3	Evaluate	Resit	3.b.ii.
15	3.b.iii.	5	Evaluate	Resit	3.b.iii.
16	4.a.	1	Remember	Actual	4.a.
17	4.b.	13	Analyze	Actual	4.b.
18	5.a.	4	Remember	Resit	5.a.
19	5.b.	10	Evaluate	Resit	5.b.
20					
21					
22					
23					
24					
25					
26					
27					
28					
29					
30					
	Total Marks	70			

Template by Tan Szu Tak

Bloom	No. Q	Marks
Remember	9	19
Understand	0	0
Apply	1	3
Analyze	1	13
Evaluate	8	35
Create	0	0

	Marks	Marks
LOTS	22	31.4%
HOTS	48	68.6%
Total	70	100.0%



Sources	No. Q	Marks	%
Actual	10	35	50.00
Resit	9	35	50.00
New	0	0	0.00

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]**

Note: Students answers may varies in terms but only the right content in the right context will be given marks. It is so that specific versions could be recalled later. **Blooms:** Remember (LOT)	[1 mark]

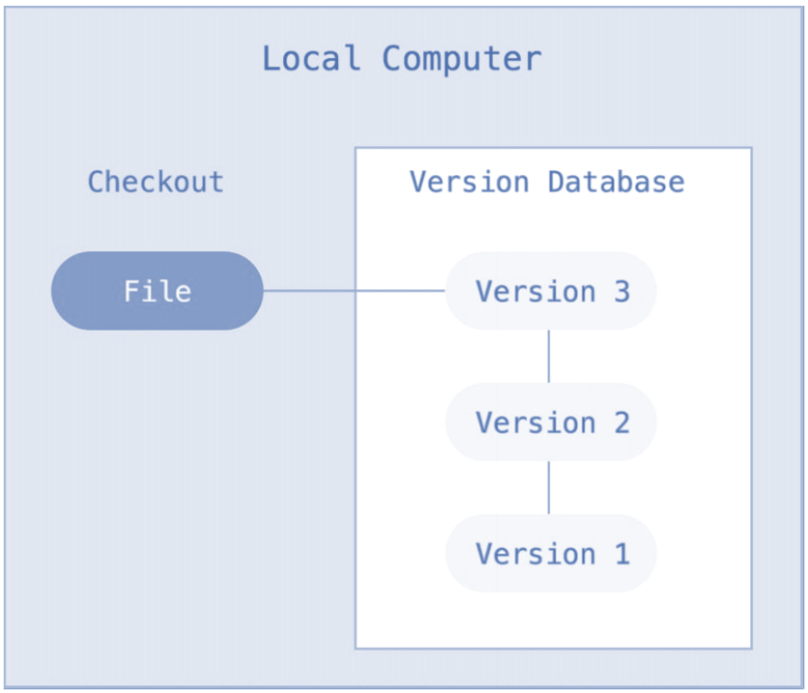
b) There are **three (3)** types of Version Control Systems (VCS) and one of them is Local Version Control Systems. List the other **two (2)** types of VCS. **[1 mark]**

Note: Students answers may varies in terms but only the right content in the right context will be given marks. - Centralised version control systems - Distributed version control systems **Blooms:** Remember (LOT)	[0.5 marks] [0.5 marks]

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

Student's solution has to satisfy the following criteria: - Illustrated local version control systems (LVCS) computer - Inside the LVCS computer contain version database - Inside version database contain at least one version - Inside the LVCS computer contain file(s) - Draw line from version database to each file(s) in the computers	 [0.5 marks] [0.5 marks] [0.5 marks] [0.5 marks] [0.5 marks]

[Turn over

- No additional computer(s) or server(s) in the illustration. Sample solution:  **Blooms:** Apply (LOT)	[0.5 marks]

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

Note: Students answers may varies in terms but only the right content in the right context will be given marks. Modified You have changed the file but have not committed it to your database yet Staged You have marked a modified file in its current version to go into your next commit snapshot.	[1 mark] [1 mark]

Committed The data is safely stored in your local database. **Blooms:** Remember (LOT)	[1 mark]

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

Note: Students answers may varies in terms but only the right content in the right context will be given marks. **Any valid three (3) features [1 mark each]**, example features are as follows: - It allows you to revert selected files back to a previous state - Revert the entire project back to a previous state - Compare changes over time - See who last modified something that might cause a problem - Who introduced an issue and when. **Blooms:** Remember (LOT)	[3 marks]

f) List **and** explain **two (2)** ways using Git to update the development environment.

[3 marks]

Note: Students answers may varies in terms but only the right content in the right context will be given marks. Git fetching - The data from remote repository only travel to the local repository. Git pulling - The data from remote repository travel to local repository and working directory	[0.5 marks] [1 mark] [0.5 marks] [1 mark]

[Turn over

Blooms: Remember (LOT)	

QUESTION 2 [Total marks: 14]

a) "Java Binary Input/Output is more efficient than Text Input/Output." Justify the statement.

[1 mark]

Note: Students answers may varies in terms but only the right content in the right context will be given marks. Because Binary Input/Output does not involve encoding or decoding the file. **Blooms:** Remember (LOT)	[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

```
1.  public class SaveStudentFile {
2.    public static void main(String[] args) {
3.      try(DataOutputStream output = new DataOutputStream(new
4.          FileOutputStream("student.dat", false))) {
5.        output.writeInt(123);
6.        output.writeUTF("Jailani");
7.        output.writeDouble(98.5);
8.        output.writeUTF("SICT");
9.      } catch (IOException e) {
10.       e.printStackTrace();
11.     }
12.   }
13. }
```

Table 1

LoadStudentFile.java

```
1.  public class LoadStudentFile {
2.    public static void main(String[] args) {
3.      try(/*TODO1*/) {
4.      /*TODO2*/
5.    } catch (IOException e) {
6.      e.printStackTrace();
7.    }
8.    }
9.  }
```

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]**

Student's code has to satisfy the following criteria: /*TODO1*/ • Create appropriate InputStream object to read a file student.dat • The InputStream object created must be appropriate to read the data in the file /*TODO2*/	[1 mark] [1 mark] [1 mark]

[Turn over

• Read the content of the file in the correct order. (Deduct 0.5 marks for each incorrect order. Maximum deduction 1 mark.) • Output all the content of the file to the console. (Deduct 0.5 marks for each content not outputted. Maximum deduction 1 mark.) Sample solution: /*TODO1*/ DataInputStream input = new DataInputStream(new FileInputStream("student.dat")) /*TODO2*/ System.out.println(input.readInt()); System.out.println(input.readUTF()); System.out.println(input.readDouble()); System.out.println(input.readUTF()); **Blooms:** Evaluate (HOT)	[1 mark]

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

```
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 4

[Turn over

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 5

i. In Table 4, implement the appropriate InputStream object at Line 8 commented with TODO 1 that will read the File object, file, in Line 6. **[3 marks]**

The student's solution should satisfy the following criteria:	
• Declare a variable with identifier, readData, with data type ObjectInputStream.	[0.5 marks]
• Create a ObjectInputStream object.	[0.5 marks]
• Create a FileInputStream object.	[0.5 marks]
• Pass the FileInputStream object as the argument when creating a ObjectInputStream object.	[0.5 marks]
• Pass the File object, file, as the argument when creating the FileInputStream object.	[0.5 marks]
• The ObjectInputStream object must be assigned to the variable.	[0.5 marks]
Example Solution: `ObjectInputStream readData = new ObjectInputStream(new FileInputStream(file));` **Blooms:** Evaluate (HOT)	

ii. In Table 4, implement the appropriate OutputStream object at Line 17 commented with TODO 2 that will append the File object, file, in Line 6. **[3 marks]**

The student's solution should satisfy the following criteria:	
• Declare a variable with identifier with data type ObjectOutputStream.	[0.5 marks]
• Create a ObjectOutputStream object.	[0.5 marks]
• Create a FileOutputStream object.	[0.5 marks]
• Pass the FileOutputStream object as the argument when creating a ObjectOutputStream object.	[0.5 marks]
• Pass the File object, file, as the first argument and Boolean value true as second argument when creating the FileOutputStream object.	[0.5 marks]
• The ObjectOutputStream object must be assigned to the variable.	[0.5 marks]
Example Solution: `ObjectOutputStream writeData = new ObjectOutputStream(new FileOutputStream(file));` **Blooms:** Evaluate (HOT)	

iii. In Table 4, 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, 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 **and** its integrity still maintained. **[3 marks]**

The student's solution should satisfy the following criteria:	
• Create a Children object.	[0.5 marks]
• Pass the arguments following the order of data when creating the Children object:	
i. Data in variable, name.	[0.5 marks]
ii. Data in variable, year.	[0.5 marks]
iii. Data in variable, month.	[0.5 marks]
iv. Data in variable, day.	[0.5 marks]
• Use the variable in Question 2.c)ii. to call the method writeObject and pass the created Children object as its argument.	

[Turn over

Example Solution:
`Children children = new Children(name, year, month, day);`
`writeData.writeObject(children);`
 	
Blooms: Evaluate (HOT)
	[0.5 marks]

QUESTION 3 [Total marks: 14]

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

Note: Students answers may varies in terms but only the right content in the right context will be given marks. A thread provides the mechanism for running a task. With Java, you can launch multiple threads from a program concurrently. **Blooms:** Remember (LOT)	[1 mark] [1 mark]

b) The following Table 6 and Table 7 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 6

StudentClient.java

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

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: 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]

The student's solution should satisfy the following criteria:	
• Create a ServerSocket object with integer value 9991 as its constructor argument.	[0.5 marks]
• Declare a ServerSocket variable and reference the created server socket object to the variable.	[0.5 marks]
• Use the variable that reference the ServerSocket object to wait for any connection request from the client and once connected get the Socket object of that connection.	[0.5 marks]
• Declare a Socket variable and reference the Socket object of that connection.	[0.5 marks]
• Create an DataInputStream object that uses the InputStream of the Socket object (using the Socket variable).	[0.5 marks]
• Declare an DataInputStream variable and reference the DataInputStream.	[0.5 marks]
• Create an DataOutputStream object that uses the OutputStream of the Socket object (using the Socket variable).	[0.5 marks]
• Declare an DataOutputStream variable and reference the DataOutputStream.	[0.5 marks]

Example Solution:
`ServerSocket serverSocket = new ServerSocket(9991);`
`Socket socket = serverSocket.accept();`	

[Turn over

```
DataInputStream input = new DataInputStream
(socket.getInputStream());
DataOutputStream output = new DataOutputStream
(socket.getOutputStream());
```

Blooms: Evaluate (HOT)

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: 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]**

The student's solution should satisfy the following criteria:

- Create a Socket object with String value 107.212.10.203 followed by integer value 9991 as its constructor argument. [0.5 marks]
- Declare a Socket variable and reference the Socket object. [0.5 marks]
- Create an DataInputStream object that uses the InputStream of the Socket object (using the Socket variable). [0.5 marks]
- Declare an DataInputStream variable and reference the DataInputStream. [0.5 marks]
- Create an DataOutputStream object that uses the OutputStream of the Socket object (using the Socket variable). [0.5 marks]
- Declare an DataOutputStream variable and reference the DataOutputStream. [0.5 marks]

Example Solution:

```
Socket socket = new Socket("107.212.10.203", 9991);
DataInputStream input = new DataInputStream
(socket.getInputStream());
DataInputStream output = new DataInputStream
(socket.getOutputStream());
```

Blooms: Evaluate (HOT)	

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 the implementation is for server application and client application respectively):

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]**

Note: Students answers may varies in terms but only the right content in the right context will be given marks.	
Server application:	
• Receive String data type using the readUTF method from DataInputStream object referenced variable declared in 3.b)i.	[0.5 marks]
• Send String data type using writeUTF method from DataOutputStream object referenced variable declared in 3.b)i.	[0.5 marks]
• The String data sent is "Abu"	[0.5 marks]
• Send integer data type using writeInt method from DataOutputStream object referenced variable declared in 3.b)i.	[0.5 marks]
• The integer data sent is 10.	[0.5 marks]
Client application	
• Send String data type using the writeUTF method from the DataOutputStream object referenced variable declared in 3.b)ii.	[0.5 marks]
• The String data sent is "20FTT9991"	[0.5 marks]
• Receive String data type using the readUTF method from DataInputStream object referenced variable declared in 3.b)ii.	[0.5 marks]

[Turn over

• Receive String data type using the readUTF method from DataInputStream object referenced variable declared in 3.b)ii.	[0.5 marks]

The order of sending and receiving data for the server and client applications are as above.
 	
Sample Solution:
Server application:
`input.readUTF();`
`output.writeUTF("Abu");`
`output.writeInt(10);`
 	
Client application:
`output.writeUTF("20FTT9991");`
`input.readUTF();`
`input.readInt();`
 	
Blooms: Evaluate (HOT)
	[0.5 marks]

QUESTION 4 [Total marks: 14]

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

Note: Students answers may varies in terms but only the right content in the right context will be given marks. Thymeleaf is a modern server-side Java template engine for both web and standalone environments. **Blooms:** Remember (LOT)	[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

```
1.  <!DOCTYPE html>
2.  <html lang="en" xmlns:th="http://thymeleaf.org">
3.  <head></head>
4.  <body>
5.  <h1 th:text="${u.id} + ' ' + ${u.name}"></h1>
6.  <p th:text="${u.percentage}"></p>
7.  <p th:text="${u.active}"></p>
8.  </body>
9.  </html>
```

Table 8

ControllerClass.java

```
1.  @Controller
2.  public class ControllerClass {
3.    @RequestMapping("/")
4.    public String home(ModelMap modelMap) {
5.      User user = new User(551, "Abu", 77.5, true);
6.      modelMap.put("u", user);
7.      return "index";
8.    }
9.  }
```

Table 9

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]

Student's Plain Old Java Object Class has to satisfy the following criteria:	
• Define the POJO class.	[0.5 marks]
• Declare four instance variables using the valid data types. [0.5 marks] each	[2 marks]
• Define constructor for POJO class with parameter variable.	[0.5 marks]
• Initialise the appropriate instance variables with the appropriate parameter variables. [0.5 marks] each	[2 marks]
• Implement the getter methods [0.5 marks] each and return data [0.5 marks] each.	[4 marks]
• Implement the setter methods [0.5 marks] each and setting its appropriate variable [0.5 marks]	[4 marks]

[Turn over

Sample solution:

```
public class User {
  int id; String name; double percentage; boolean active;

  public User(int id, String name, double percentage, boolean
active) {
    this.id = id;
    this.name = name;
    this.percentage = percentage;
    this.active = active;
  }
  public int getId() {return id;}
  public void setId(int id) {this.id = id;}
  public String getName() {return name;}
  public void setName(String name) {this.name = name;}
  public double getPercentage() {return percentage;}
  public void setPercentage(double percentage) {
    this.percentage = percentage;}
  public boolean isActive() {return active;}
  public void setActive(boolean active) {this.active = active;}
}
```

Blooms: Analyse (HOT)

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]

Note: Students answers may varies in terms but only the right content in the right context will be given marks.

create creates the database tables, destroying previous data.	[1 mark]
create-drop Drop the database tables when the application is stopped.	[1 mark]
update update the database tables.	[1 mark]
validate validate the database tables, makes no changes to the database.	[1 mark]
Blooms: Remember (LOT)	

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

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

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 to be a Database entity that will be used by Hibernate to create the Database table (Table 11) with its field and constraints accordingly. **[10 marks]**

[Turn over

Student's code has to satisfy the following criteria:	
• Declare a String instance variable with identifier id with @Id annotation.	[0.5 marks]
• Declare a String instance variable with identifier name with @NotNull annotation.	[0.5 marks]
• Declare a boolean instance variable with identifier active.	[0.5 marks]
• Declare an int instance variable with identifier age.	[0.5 marks]
• Define a constructor with String, String, int and boolean parameter variables (no particular order).	[0.5 marks]
• In the constructor block:	
○ Initialise instance variable id with its appropriate parameter variable.	[0.5 marks]
○ Initialise instance variable name with its appropriate parameter variable.	[0.5 marks]
○ Initialise instance variable active with its appropriate parameter variable.	[0.5 marks]
○ Initialise instance variable age with its appropriate parameter variable.	[0.5 marks]
• Define a setter method for instance variable id.	[0.5 marks]
• Define a setter method for instance variable name.	[0.5 marks]
• Define a setter method for instance variable active.	[0.5 marks]
• Define a setter method for instance variable age.	[0.5 marks]
• Define a getter method for instance variable id.	[0.5 marks]
• Define a getter method for instance variable name.	[0.5 marks]
• Define a getter method for instance variable active.	[0.5 marks]
• Define a getter method for instance variable age.	[0.5 marks]
• All instance variables are declared with private access modifier, the constructor and setter and getter methods are defined with public access modifier.	[0.5 marks]
• Defined Entity and Table with table name annotations on top of the class definition.	[0.5 marks]
• Defined Entity and Table with table name annotations on top of the class definition.	[0.5 marks]
Sample solution:	

```
@Entity
@Table(name = "students")
public class Student {
  @Id
  private String id;

  @NotNull
  private String name;
  private int age;
  private boolean active;

  public Student(String id, String name, int age, boolean active) {
    this.id = id;
    this.name = name;
    this.age = age;
    this.active = active;
  }

  public String getId() { return id; }
  public void setId(String id) { this.id = id; }
  public String getName() { return name; }
  public void setName(String name) { this.name = name; }
  public int getAge() { return age; }
  public void setAge(int age) { this.age = age; }
  public boolean isActive() { return active; }
  public void setActive(boolean active) { this.active = active; }
}
```

Blooms: Create (HOT)

[END OF MARKING SCHEME]