ڤوليتيكنيك بروني
POLITEKNIK BRUNEI

END-OF-SEMESTER EXAMINATION (RESIT)

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:

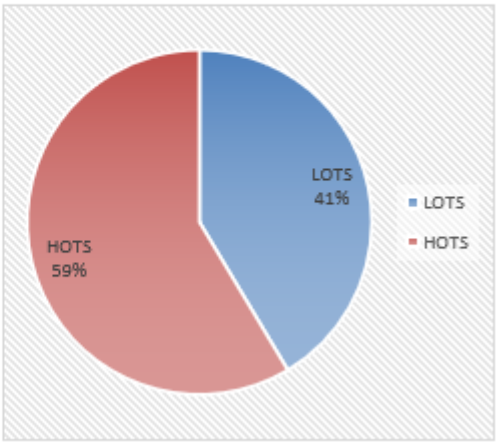
- Students have to answer ALL questions
- Students have 2 Hours to attempt this paper unless otherwise specified.
- The total mark for this paper is **70**. The number of marks for each question or part question is shown in brackets.
- The answers to the paper are shown in blue colour fonts.
- Keywords/ 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
1	1.a.	1	Remember
2	1.b.	1	Remember
3	1.c.	4	Apply
4	1.d.	3	Remember
5	1.e.	2	Remember
6	1.f.	3	Remember
7	2.a.	4	Understand
8	2.b.i.	3	Evaluate
9	2.b.ii.	3	Evaluate
10	2.b.iii.	4	Evaluate
11	3.a.	2	Remember
12	3.b.i.	4	Evaluate
13	3.b.ii.	3	Evaluate
14	3.b.iii.	5	Evaluate
15	4.a.	4	Remember
16	4.b.	1	Remember
17	4.c.i.	4	Evaluate
18	4.c.ii.	5	Evaluate
19	5.a.	4	Remember
20	5.b.	10	Evaluate
21			
22			
23			
24			
25			
26			
27			
28			
29			
30			
	Total Marks	70	

Template by Tan Szu Tak

Bloom	No. Q	Marks
Remember	9	21
Understand	1	4
Apply	1	4
Analyze	0	0
Evaluate	9	41
Create	0	0

	Marks	Marks
LOTS	29	41.4%
HOTS	41	58.6%
Total	70	100.0%



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 vary 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]

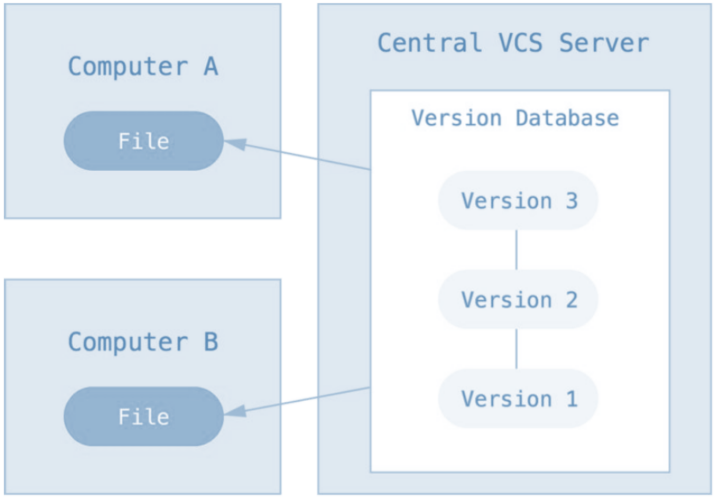
b) There are three (3) types of Version Control Systems (VCS) and one of them is Centralised Version Control Systems. List the other **two (2)** types of VCS. **[1 mark]**

Note: Students answers may vary in terms but only the right content in the right context will be given marks. - Distributed Version Control Systems - Local Version Control Systems **Blooms:** Remember (LOT)	[0.5 marks] [0.5 marks]

c) Illustrate the basic structure of Centralised Version Control Systems. **[4 marks]**

Student's solution has to satisfy the following criteria: - Illustrated centralise version control systems (CVCS) server - Inside the CVCS server contain version database - Inside version database contain any amount of version(s)	 [0.5 marks] [0.5 marks] [0.5 marks]

[Turn over

- Illustrated at least two computers with any identifier (Example, Computer A and Computer B) and must not be in the CVCS server [0.5 marks for each computer) - Inside each computer contain file(s) [0.5 marks for each computer] - Draw arrow from version database to each file(s) in the computers Sample solution:  **Blooms:** Apply (LOT)	[1 mark] [1 mark] [0.5 marks]

d) 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]

e) Explain what happens when you clone a Git repository. **[2 marks]**

Note: Students answers may varies in terms but only the right content in the right context will be given marks.	
The data from the remote repository travel to 2 areas • Working directory • Local repository	[1 mark] [0.5 marks] [0.5 marks]
Blooms: Remember (LOT)	

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

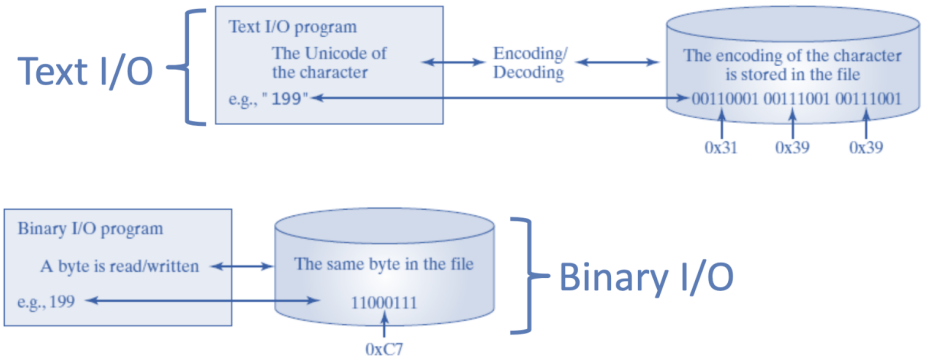
[3 marks]

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.	[0.5 marks] [1 mark]
Git pulling - The data from remote repository travel to local repository and working directory	[0.5 marks] [1 mark]
Blooms: Remember (LOT)	

[Turn over

QUESTION 2 [Total marks: 14]

a) Explain in detail the difference between Java Binary Input / Output **and** Java Text Input / Output in terms of writing and reading to/from a file. (Note: You may illustrate using diagram or use an example to further enhance you explanation) **[4 marks]**

Note: Students answers may varies in terms but only the right content in the right context will be given marks.	
Java Binary Input / Output It read / written data as the byte representation of the data and the same byte are read / written to the file.	[1 mark] [1 mark]
Java Text Input / Output It read / write data as the Unicode of the character and it encode or decode data when write or read the file accordingly.	[1 mark] [1 mark]
Example Diagram: 	
Blooms: Understand (LOT)	

b) 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]**

The student's solution should satisfy the following criteria:	
• Declare a variable with identifier, readData, with data type DataInputStream.	[0.5 marks]
• Create a DataInputStream object.	[0.5 marks]
• Create a FileInputStream object.	[0.5 marks]

[Turn over

• Pass the FileInputStream object as the argument when creating a DataInputStream object. • Pass the File object, file, as the argument when creating the FileInputStream object. • The DataInputSteam object must be referenced to the variable. Example Solution: `DataInputStream readData = new DataInputStream(new FileInputStream(file));` **Blooms:** Evaluate (HOT)	[0.5 marks] [0.5 marks] [0.5 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]**

The student's solution should satisfy the following criteria: • Declare a variable with identifier with data type DataOutputStream. • Create a DataOutputStream object. • Create a FileOutputStream object. • Pass the FileOutputStream object as the argument when creating a DataOutputStream object. • Pass the File object, file, as the first argument and Boolean value true as second argument when creating the FileOutputStream object. • The DataOutputStream object must be referenced to the variable. Example Solution: `DataOutputStream writeData = new DataOutputStream(new FileOutputStream(file, true));` **Blooms:** Evaluate (HOT)	 [0.5 marks] [0.5 marks] [0.5 marks] [0.5 marks] [0.5 marks] [0.5 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 2.b)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]**

The student's solution should satisfy the following criteria:	
• Use the variable in Question 2.b)ii. to call the method writeUTF and pass the variable, name, as its argument.	[0.5 marks]
• Use the variable in Question 2.b)ii. to call the method writeInt and pass the variable, year, as its argument.	[0.5 marks]
• Use the variable in Question 2.b)ii. to call the method writeInt and pass the variable, month, as its argument.	[0.5 marks]
• Use the variable in Question 2.b)ii. to call the method writeInt and pass the variable, day, as its argument.	[0.5 marks]
• The following is the order of data needs to be written into the file.	
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]

Example Solution:
`writeData.writeUTF(name);`
`writeData.writeInt(year);`
`writeData.writeInt(month);`
`writeData.writeInt(day);`
 	
Blooms: Evaluate (HOT)	

[Turn over

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 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

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]

The student's solution should satisfy the following criteria:
• Create a ServerSocket object with integer value 9991 as its constructor argument.
• Declare a ServerSocket variable and reference the created server socket object to the variable.
• 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.
• Declare a Socket variable and reference the Socket object of that connection.
• Create an DataInputStream object that uses the InputStream of the Socket object (using the Socket variable).
• Declare an DataInputStream variable and reference the DataInputStream.
• Create an DataOutputStream object that uses the OutputStream of the Socket object (using the Socket variable).
• Declare an DataOutputStream variable and reference the DataOutputStream.
 	
Example Solution:
`ServerSocket serverSocket = new ServerSocket(9991);`
`Socket socket = serverSocket.accept();`
`DataInputStream input = new DataInputStream (socket.getInputStream());`
`DataOutputStream output = new DataOutputStream (socket.getOutputStream());`
 	
Blooms: Evaluate (HOT)
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]

[Turn over

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]**

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 DatatInputStream object that uses the InputStream of the Socket object (using the Socket variable).	[0.5 marks]
• Declare an DatatInputStream variable and reference the DatatInputStream.	[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 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 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]
• Receive String data type using the readUTF method from DataInputStream object referenced variable declared in 3.b)ii.	[0.5 marks]

[Turn over

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) Differentiate between Uniform Resource Locator (URL) **and** Uniform Resource Identifier (URI). **[4 marks]**

Note: Students answers may varies in terms but only the right content in the right context will be given marks. URL In general a URL is an address that includes a method for locating a resource, with a protocol like HTTP or FTP. URI Is the resource location where no host name and protocol are included. **Blooms:** Remember (LOT)	 [1 mark] [1 mark] [1 mark] [1 mark]

b) Briefly explain the concept of Model View Controller. **[1 mark]**

Note: Students answers may varies in terms but only the right content in the right context will be given marks. A design pattern for implementing user interfaces on computers. **Blooms:** Remember (LOT)	[1 mark]

c) The following Table 4 is a Java class called MainController.java.

MainController.java

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

Table 4

i. Implement a method in Table 4 to handle Uniform Resource Identifier (URI) "/" where it will respond with a literal String "NS4307 Network Programming". (Note: You are not required to rewrite the whole class.) **[4 marks]**

The student's solution should satisfy the following criteria:	
• Implements a method with correct syntax,	
○ With any identifier	[0.5 marks]
○ No parameter	[0.5 marks]
○ String return data type	[0.5 marks]
• Implements return "NS4307 Network Programming" inside the method block.	[0.5 marks]
• Implements @RequestMapping annotation with value "/"	[0.5 marks]
• The @RequestMapping annotation is on top of the method definition.	[0.5 marks]
• Implements @ResponseBody annotation	[0.5 marks]
• The @ResponseBody annotation is on top of the method definition	[0.5 marks]

Sample solution:
`@RequestMapping(value="/")`
`@ResponseBody`
`public String home() {`	

[Turn over

`return "NS4307 Network Programming";`
`}`
 	
Blooms: Evaluate (HOT)	

ii. Implement a method in Table 4 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]**

The student's solution should satisfy the following criteria:
• Implements a method with correct syntax,
  ○ With any identifier
  ○ Declare parameter variable with identifier, message, of String data type.
  ○ Implements @PathVariable annotation.
  ○ The @PathVariable annotation is at the front of the parameter variable, message, declaration
  ○ String return data type
• Implements return value in the parameter variable, message, inside the method block.
• Implements @RequestMapping annotation with value "/{message}"
• The @RequestMapping annotation is on top of the method definition.
• Implements @ResponseBody annotation
• The @ResponseBody annotation is on top of the method definition
 	
Sample solution:
`@RequestMapping(value="/{message}")`
`@ResponseBody`
`public String message(@PathVariable String message) {`
`return "message";`
`}`
Blooms: Evaluate (HOT)
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 marks]
	[0.5 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]

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 5 is a Java class called Student.java and Table 6 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.

[Turn over

Student.java

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

Table 5

```
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 6

Rewrite the Plain Old Java Object class, Student.java (Table 5) by making the class to be a Database entity that will be used by Hibernate to create the Database table (Table 6) with its field and constraints accordingly. **[10 marks]**

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] [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. • 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;
```

[Turn over

```
    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]