Software Engineering Departmenta of the College of Engineering

Final Examination for the Course Advanced Programming

Examination Date: 28th November 2025 **Academic Year: 2025/26**
Maximum score: 50% **Semester: I**
Time Allowed: 2 hours **Target Group: 3rd Year**

Tick an 'X' mark in the box where it applies you.

Freshman	Applied Science		Engineering	
Regular (2nd & above):	Undergraduate		Postgraduate	
Continuing Education	Undergraduate		Postgraduate	

Course instructor(s) name(s): ☐ Dr. Tilahun Melak

Student Name: _____ ID. No: _____

General Directions

- Write your name (including your grandfather's name) and ID No. both on the question paper and answer sheet (if any).
- The exam has two parts and 10 questions, attempt to answer all of them.
- Please read the instructions carefully and attempt all the questions.
- It is strictly forbidden to bring mobile phones and any electronic devices into the examination room
- Do not use pencil and red pen.
- Do not detach the answer sheet from the question paper.
- Cheating automatically nullifies your total scores.

For Instructor's Use Only

Question Type/ No.	Mark Allotted	Mark Scored
Part I	18 pts	
Part II	32 pts	
Total	**50 pts**	

PART I

Write programs for the following questions based on each instruction

1. Questions from A-D are about LinkedList of ***Top IT Companies***. Read each question carefully and write appropriate answers. ***(5pts)***

   A. Write a program to create a new LinkedList of ***Top IT Companies***, add companies (‘Microsoft’, ’IBM’, ’Oracle’, ’HPE’) and print out the collection.

Answer

```
import java.util.*;
public class TestMid
{
public static void main (String [] args){
ArrayList <String> al= new ArrayList <String> ();
al.add("Microsoft");
al.add("IBM");
al.add("Oracle");
al.add("HPE");
System.out.println(al);
}}
```

   B. Modify the program to insert ‘SAP’ at the second position and ‘Infosys’ at the last position of the list.

Answer

```
import java.util.*;
public class TestMid
{
public static void main (String [] args){
ArrayList <String> al= new ArrayList <String> ();
al.add("Microsoft");
```

```
al.add("IBM");
al.add("Oracle");
al.add("HPE");
al.add(2,"SAP")
al.add(5,"Infosys")
System.out.println(al);
}}
```

C. Modify the program to shuffle elements in the ArrayList

Answer

```
import java.util.*;
public class TestMid
{
public static void main (String [] args){
ArrayList <String> al= new ArrayList <String> ();
al.add("Microsoft");
al.add("IBM");
al.add("Oracle");
al.add("HPE");
al.add(2,"SAP")
al.add(5,"Infosys")
System.out.println(al);
Collections.shuffle(al);
System.out.println(al);
}}
```

D. Modify the program to reverse elements in the ArrayList

Answer

```
import java.util.*;
public class TestMid
```

```
{
public static void main (String [] args){
ArrayList <String> al= new ArrayList <String> ();
al.add("Microsoft");
al.add("IBM");
al.add("Oracle");
al.add("HPE");
al.add(2,"SAP")
al.add(5,"Infosys")
Collections.shuffle(al);
System.out.println(al);
Comparator r=Comparator.reverseOrder();
Collections.sort(al,r);
System.out.println(al);
}}
```

2. Convert the following class into thread by extending Thread class. The thread will print all the odd numbers up to its bound: ***(3pts)***

```
public class PrintOdds {
    private int bound;
    public PrintOdds(int b) {
        bound = b;
    }

    public void print() {
        for (int k = 1; k < bound; k+=2)
            System.out.println(k);
    }
```

Answer

```
public class PrintOdds implements Runnable {
```

```
private int bound;
public PrintOdds(int b) {
 bound = b;
}

public void print() {
  for (int k = 1; k < bound; k+=2)
    System.out.println(k);
}

public void run() {
  print();
}
```

3. Given the following

Deriver=*" com.mysql.jdbc.Driver"*

Database url= *" jdbc:mysql://localhost:3306/**world** "* ***(6pts)***

A. Write a full jdbc program to create a table called ***city*** in the ***world*** database with columns ID (primary key), Name, CountryCode, District, Population.

Answer

```
import java.sql.*;
class jdbc{
public static void main (String [] args){
try{
Class.forName(" com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/Stockmarket",
                  "root","root");
Statement stmt =con.createStatement();
Stmt.excuteUpdate("Create Table city (ssn varchar(20), cust-name varchar(20),
address varchar(20), primary key(ssn)");
con.close();
     }
      catch(Exception e)
      {
          e.printStackTrace();
      }
```

```
}

}
```

}

B. Write a jdbc code to add the following record to the city table.

ID	Name	CountryCode	District	Population
756	Addis Ababa	Eth	Addis Ababa	2495000

Answer

```
stmt.executeUpdate("insert into city values ('756',' Addis Ababa',' Eth','
Addis Ababa', '2495000')");
```

C. Assume that you added more rows to the table. Write a jdbc code to display all cities.

Answer

```
ResultSet rs=stmt.executeQuery("select * from City");
 System.out.println("ID\t\tName\t CountryCode \t District \t
Population ");
 System.out.print("---------------------------------------\n");
while(rs.next())
{
System.out.println(rs.getString(1)+"\t"+rs.getString(2)+"\t"+rs.g
etString(3)+"\t"+rs.getString(4) "+rs.getString(5));
}
System.out.println("---------------------------------------");
```

D. Write a jdbc code to delete the following record from city table.

ID	Name	CountryCode	District	Population
757	Dire Dawa	Eth	Dire Dawa	164851

Answer

```
stmt.executeUpdate("delete from student where id='757");
```

4. Given that port number=1234 and host name="jtutorial.edu.et" ***(4pts)***

A. Write a client side code to connect with the specified port number for communication.

Answer

```
try {
        Socket sock = new Socket("jtutorial.edu.et ", 1234);
    } catch ( UnknownHostException e ) {
        System.out.println("Can't find host.");
    } catch ( IOException e ) {
        System.out.println("Error connecting to host.");
    }
```

B. Write a server side code to accept a connection from the client.

Answer

```
try {
        ServerSocket listener = new ServerSocket( 1234 );

        while ( !finished ) {
            Socket client = listener.accept( );
…
            }
```

PART II

Write the appropriate brief answers for the following questions

1. Explain why and when we use **List**, **Set** and **Map** with examples. ***(3pts)***

Answer

List

- **Why/When:** When you need an ordered collection that allows duplicates and provides positional access (index-based).
- **Example:**

```
List<String> students = new ArrayList<>();
students.add("Alem");
students.add("Sara");
```

```
students.add("Alem");   // duplicates allowed
```

Set

- **Why/When:** When you need a collection of **unique** elements with **no duplicates**; order is not important (unless using LinkedHashSet or TreeSet).
- **Example:**

```
Set<String> ids = new HashSet<>();
ids.add("ID01");
ids.add("ID02");
ids.add("ID02");  // ignored (duplicate)
```

Map

- **Why/When:** When storing **key–value pairs**, e.g., dictionaries, lookups.
- **Example:**

```
Map<String, Integer> scores = new HashMap<>();
scores.put("Alem", 90);
scores.put("Sara", 85);
```

2. Discuss Benefits of Java Collections Framework by at least mentioning three of them. ***(3pts)***

Answer

1. **Reusability** – provides ready-made data structures and algorithms.
2. **Efficiency** – optimized implementations of lists, sets, maps.
3. **Consistency** – common interfaces (Collection, List, Set, Map).
4. **Interoperability** – can easily switch between different collection types.
5. **Reduces coding effort** – built-in methods for searching, sorting, etc.

3. A thread can be in certain state at a given time. Discuss the detail life cycle of a thread using a diagram. ***(3pts)***

Answer

A thread moves through the following states:

1. **New** – thread object created but not started.

2. **Runnable** – after calling `start()`, eligible to run.
3. **Running** – executing on CPU.
4. **Blocked/Waiting** – waiting for I/O, lock, or other threads.
5. **Timed Waiting** – waiting for a specified time (e.g., `sleep(1000)`).
6. **Terminated** – execution completed or stopped due to error.

4. Discuss thread synchronization with examples. (**3pts**)

Answer

- Synchronization prevents **race conditions** when multiple threads access shared data.
- Uses the **synchronized** keyword to lock a code block or method.

Example:

```
class Counter {
    private int count = 0;

    public synchronized void increment() {
        count++;
    }
}
```

- Only one thread can execute `increment()` at a time → prevents inconsistent values.

5. Discuss the general architecture of Remote Method Invocation (RMI). ***(3pts)***

Answer

RMI architecture has:

1. **Client** – invokes remote methods.
2. **Server** – hosts remote objects.
3. **Stub** (client-side proxy) – forwards calls to server.
4. **Skeleton** (server-side proxy; prior to Java 1.2) – dispatches requests.
5. **RMI Registry** – binds and looks up remote objects.
6. **RMI Transport Layer** – handles communication over TCP/IP.

6. List the eight Steps for Developing an RMI System. ***(3pts)***

Answer

1. Define the **remote interface**.
2. Implement the **remote class**.
3. Compile the files.
4. Generate stubs (older versions; now automatic).
5. Start the **RMI Registry**.
6. Start the **server program** to bind objects.
7. Write the **client program**.
8. Run the **client** to access remote methods.

7. Describe how servlets and containers support dynamic web content, session management, persistence, and security. (2pts)

Answer

- **Dynamic content:** Servlet generates HTML/JSON at runtime.
- **Session management:** Cookies, URL rewriting, `HttpSession`.
- **Persistence:** Database connections through JDBC inside servlets.
- **Security:** Authentication, authorization, HTTPS, container-managed security.

8. Identify the advantages of servlets over CGI, including performance, portability, and robustness. (2pts)

Answer

1. **Better performance** – servlet stays in memory; CGI creates a new process per request.
2. **Portability** – runs on any Java-enabled server.
3. **Robustness** – automatic garbage collection, strong typing.
4. **Scalability** – efficient handling of many requests (multithreading).

9. Differentiate between the javax.servlet and javax.servlet.http packages. (2pts)

Answer

- **javax.servlet:**
  - Generic servlet classes and interfaces (Servlet, Filter, RequestDispatcher).
  - Protocol-independent.
- **javax.servlet.http:**

- HTTP-specific classes (HttpServlet, HttpServletRequest, HttpServletResponse).
- Adds session and cookie handling.

10. Explain different ways to create a servlet and why extending the HttpServlet class is commonly used. (2pts)

Answer

Ways:

1. Extend **GenericServlet**.
2. Extend **HttpServlet**.
3. Implement the **Servlet** interface directly.

Why HttpServlet is commonly used:

- Provides built-in methods (`doGet`, `doPost`).
- Handles HTTP-specific features (sessions, cookies).
- Minimal code required.

11. Define purpose and architecture of JavaBeans. (2pts)

Answer

Purpose:

- Reusable, platform-independent software components.

Architecture:

- A Java class with:
  - No-argument constructor
  - Private properties
  - Public getter/setter methods
  - Serializable

Beans can be visually manipulated in IDEs.

12. Explain properties, events, and persistence. (2pts)

Answer

- **Properties:** Attributes of a bean accessed via getters/setters.
- **Events:** Beans use the event model to interact (listeners, callbacks).
- **Persistence:** Beans save and restore their state using serialization.

13. Describe customization and component reuse. (2pts)

Answer

- **Customization:**
  - Configure bean properties through editors or code.
  - Allows developers to modify behavior without altering source code.
- **Component Reuse:**
  - Beans can be assembled into larger applications.
  - Promotes modularity and faster development.