JAVA

Amith. N
2310390022

▷ Explain with an example various steps involved in creating a GUI.

The various steps involved in creating a GUI are,

① Declare for GUI component types,

Create a variable for each of the GUI components that you want to use in GUI.

Ex: If you want to create a button, you would create a variable of type JButton.

② Call the appropriate constructor with suitable arguments.

Once you have declared your variables, you need to create instances of GUI components that you want to use. To do this you need to call the appropriate constructor with suitable arguments.

Ex: Create a button with text "Click me!", you would call the following constructor.

```
JButton button = new JButton("Click me!");
```

③ Add these components to the container (Applet, Frame & Panel)

Once you have created your GUI components, you need to add them to a container. A container is a GUI component that can hold other GUI components. The most common containers are Applet, Jframe and Panel.

To add a GUI component to a container, you can use the add() method.

Ex: To add button that you created in previous step to frame you would call the following code,

```
Jframe frame = new JFrame();
frame.add(button);
```

④ Add the appropriate listeners to these components.

A listener is a java object that can respond to events generated by GUI components. For example, you can add a listener to a button to respond to when button is clicked.

To add a listener to a GUI component, you can use addActionListener() method.

```
button.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        //Handle the button click event here
    }
});
```

⑤ Override the listener methods & handle the event generated by them:

Once you have added a listener to a GUI component, you need to override the listener methods & handle the events generated by them.

Ex: To handle the button click event, you would override the actionPerformed() method.

Ex:

```
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;

public class MyGUI {
    public static void main(String[] args)
    {
        // Create the frame
        JFrame frame = new JFrame();
        // Create the button
        JButton button = new JButton("Click me!");
        // Add button to frame
        frame.add(button);
        // Add a listener to the button
        button.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                // Handle the button click event here
                System.out.println("Button clicked!");
            }
        });
        // set frame's size & visibility.
```

```
        frame.setSize(300, 300);
        frame.setVisible(true);
    }
}
```

2) What is the difference between a database schema and a database state?

Database Schema	Database State
* The database schema changes very frequently.	* Database state changes every time the database is updated.
* Schema is also called intension.	* State is also called extension.
* Database schema represent overall design of database.	* Database state represent current state of data in database.
* Initially when defining a database, only database schema is specified.	* Initially, when defining a database, database state is empty state.

3) What are various categories of Data Models?

The various categories of Data models are,

① Conceptual data model: It represents high-level view of data without specifying any implementation details.

* They are used to understand the business requirements & to design logical data model.
* Concise description of data requirements.
* Includes entity types, relationships, constraints.
* Meet data & functional requirements.

② Logical data model:

* It represents data structure & relationships between the data entities, without specifying any physical implementation details.

* They are used to design the database schema of & to implement physical data model.

* Implementation of database.

* Use representational data model (relational, object).

* Choose a specific DBMS (Oracle, My SQL, Oracle...).

③ Physical data model:

* Internal storage structure, access paths etc.

* It represents specific implementation of data structure & relationships, including data types, storage structure & access methods.

* They are used to create & manage database.

④ Relational data model:

* This is most common type of data model, & is used by most commercial database systems.

* It stores data in tables, which are made up of rows & columns.

⑤ Hierarchical data model:

* This type of data model stores data in a tree-like structure, with parent child relationships between data entities.

⑥ Network data model:

* This type of data model is similar to hierarchical data model, but it allows for complex relationships between the data entities.

⑦ Object-oriented data model:

* This type of data model stores data in objects, which are self-contained entities that contain both data & behaviour

⑧ NoSQL data model:

* This type of data model is designed to handle large and complex datasets that are different to store & manage.

4) What is Serialization? Give one example for Serialization?

Serialization is the process of writing the state of an object to a byte stream. This is usefull when you want to save the state of your program to a persistent storage area, such as a file.

Ex:

```
import java.io.*;
public class SerializationDemo {
    public static void main(String args[])
    {
      // Object serialization
      try {
          MyClass objects = new Myclass("Hello", -7, 2.7e10);
          System.out.println("Object1:" + object1);
          FileOutputStream fos= new FileOutputStream("serial");
        ObjectOutputStream oos = new ObjectOutputStream(fos);
        oos.WriteObject(object1);
        oos.flush();
        oos.close();
      }
      catch(IOException e){
          System.out.println("Exception during serialization:"+e);
          System.exit(0);
      }
// Object deserialization
try {
    MyClass object2;
    FileInputStream fis= new FileInputStream("serial");
    ObjectInputStream ois = new ObjectInputStream(fis);
    object2 = (MyClass)ois.readObject();
    ois.close();
    System.out.println("object2:" + object);
}
```

```
        Catch (Exception e){
            System.out.println("Exception during deserialization:"+e);
            System.exit(0);
            }
    }
}
Class MyClass implements Serializable
    {
     String s;
     int i;
     double d;
     public MyClass(String s, int i, double d)
          {
            this.s=s;
            this.i=i;
            this.d=d;
          }
         public String toString()
           {
            return "s:"+s+";i="+i+";d="+d;
           }
         }
```

Output:

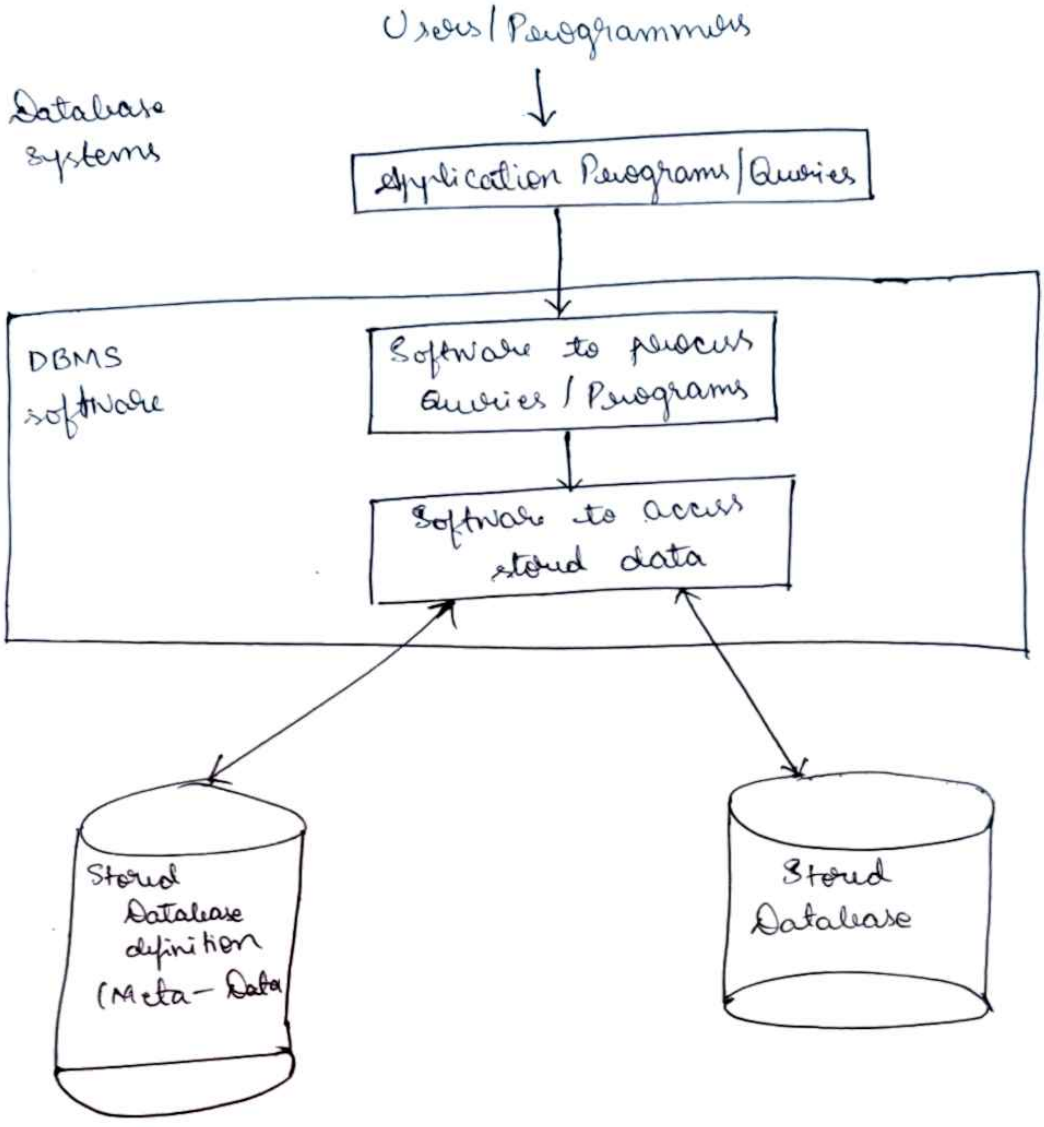
Object 1 : s= Hello; i= -7; d = 2.7E10;

Object 2 : s= Hello; i= -7; d = 2.7E10;

5) Differentiate file processing & database approach.

Feature	File processing	Database approach
* Data storage	* Data is stored in files.	* Data is stored in database.
* Data organization	* Data is typically organised by entity, with each file containing data for a single entity.	* Data is typically organised by entity with each other table representing a different entity. The tables are linked together using relationships.
* Data access	* Data is accessed sequential- tially one file at a time.	* Data can be accessed sequentially / randomly, depending on the database type.
* Data integrity	* Data integrity is difficult to maintain, as data is stored in multiple files.	* Data integrity is easier to maintain as data is stored in single database.
* Scalability	* File processing is not scalable as it can be difficult to manage large number of files.	* Database approach is more scalable, as it can handle large volumes of data.
* Security	* File processing offers limited security, as data is often stored in plain text files.	* Database approach offers better security, as data can be encrypted & access to the database can be controlled.

6) Explain database system environment with a block diagram.



Database System Environment

* Users / Programmers:

These are the people who use the database system to store retrieve & manage data.

* Database Systems:

This is the software that manages the database. It provides a layer of abstraction between users & physical storage of data.

* Application programs / queries:

These are the programs that users use to interact with the database system. They can be written in any programming language, but they must use the database system's API to communicate with database.

* DBMS software:

This is the core software component of database system. It is responsible for managing database schema, storing and retriving data & enforcing data integrity constraints.

* Software to access stored data:

This software is responsible for accessing & manipulating the data stored in database.

* Stored Database Definition:

This is the information about database schema, such as table, columns & relationships between the tables.

* Stored Database:

This is actual data stored in database.

⇒ Explain the different ways of handling the exception in Java.

The process of handling the abnormal situation that is occured is known as Exception handling.

There are two main ways of handling exceptions in Java:

① Try-Catch block:

Syntax:

```
try
{
  ...
  // statements
}
catch (Declare Throwable type Ref variable)
{
  // statements;
}
```

Try-catch block is most common way to handle exceptions. It allows you to specify a block of code to be executed, if an exception occurs. Catch block specifies type of exception that you want to handle.

Try: The statements responsible for exception must be written inside try block.

When exception occurs a throwable type object is created & reference of object is thrown to catch block.

Catch: Catch block is used to catch reference of the object thrown by try.

② Throws Keyword:

Syntax:

```
[modifier] return type  method-name  throws excpt1,...
{
  //statements;
}
```

Throws is a keyword which is used to declare an exception. Throws keyword informs compiler that method will throw exception object to its caller when exception is occured.

Ex:

```
import java.io.*;
public class Exception;
{
  public FileOutStream CreateFile (String log, String name)
        throws FoundException
        {
          return new FileOutputStream (log + name);
        }
}
```