DATE

Assignment

a.1) Explain Delegation Event Model in brief.

→ Modern approach to handling events is based on delegation event model, which defins standard and mechanisms to generate and process events.

→ The concept is that: a source generates an event and sends it to one or more listeners. listener simply waits until it receives an event and once event is received, the listener processes the event and then returns.

→ The advantage of this design is that application logic that processes events is cleartynly seperated from user interface logic that generates those events. UI element is able to delegate the processing of an event to a seperate piece of code.

→ Here, listener must register with a source in order to receive an event notification. Iso that notifications are sent only to listeners that want to receive them.

a.2) Mention the basic event handling mechanism in AWT.

→

PAGE

DATE

→ Event handling is the mechanism that controls the event and decides what should happen if an event occurs. This mechanism have the code which is known as event handler that is executed when event occurs.

→ Java uses Delegation Event model to handle the events

Delegation Event model has following key participants namely.

- Source
  - is an Object on which event occurs.
- Listener
  - is also event handler. is responsible for generating response of event.

Steps in event handling

- User click button and event is generated.
- Now object of concerned event class is created and information about source and event get populated with in same object
- Event object is forwarded to the method of registered listener class.

PAGE

DATE

- method is now executed and returns

DATE

Q.3) Explain BorderLayout with example.

→ BorderLayout is the default layout for window objects such as JFrame, JWindow, JDialog, e.t.c. It arranges components in five regions: North, south, east and west and center.

Constructors :-

i) BorderLayout()
Construct a new borderlayout with no gaps between the components.

ii) BorderLayout(int, int)
Construct a border layout with the specified gaps between the components.

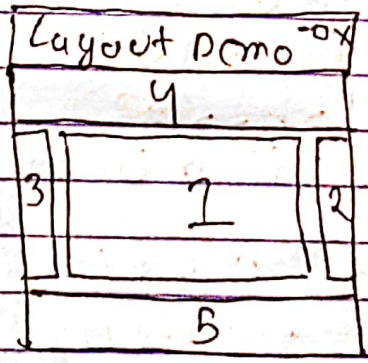
Eg :-

```
import java.awt.*;
Public class BorderLayoutDemo5
  Public BorderLayoutDemo()5
    Frame f =new Frame("Layout
                        Demo");
    f.setLayout(new BorderLayout());
    Button b1=new Button("1");
    Button b2=new Button("2");
    Button b3=new Button("3");
    Button b4=new Button("4");
    Button b5=new Button("5");
```

PAGE

DATE

```
f.add(b1, BorderLayout.CENTER);
f.add(b2, BorderLayout.EAST);
f.add(b3, BorderLayout.WEST);
f.add(b4, BorderLayout.NORTH);
f.add(b5, BorderLayout.SOUTH);
f.setSize(200,200);
f.setVisible(true);
}
Public static void main(String[]
    args){
new BorderLayoutDemo();
}
}
```



Q.4) Explain CardLayout with example.

→ CardLayout class manages the components in such a way that only one component is visible at a time. It treats each component as a card in the container. Only one card is visible at a time, and container acts as stack of

PAGE

DATE

cards.

Constructor:-

1) CardLayout()

→ It is used to create a new card layout with gaps of size is zero.

2) CardLayout(int horizontalgap, int verticalgap)

→ It is used to create a new CardLayout class with the specified horizontal and vertical gaps.

Eg:-

```
import java.awt.*;
Public class CardLayoutDemo{
  CardLayout card;
  Button b1,b2,b3;
  CardLayoutDemo(){
   Frame frame=new Frame();
    card=new CardLayout(40,30);
    frame.setLayout(card);
    b1=new Button("one");
    b2=new Button("two");
    b3=new Button("three");
    b1.
    b1.addActionListener(new
                ActionListener(){
      Public void actionperformed
                (ActionEvent e)
      {
```

PAGE

```
            card.next(frame);
        }
    });
    b2.addActionListener(new ActionListener(){
        Public void actionPerformed
            (ActionEvent e){
            card.next(frame);
        }
    });
    b3.addActionListener(new Ac-
        tionListener(){
        Public void actionPerformed
            (ActionEvent e){
            card.next(frame);
        }
    });
    frame.add("a",b1);
    frame.add("a",b2);
    frame.add("c",b3);
    frame.setVisible(true);
    frame.setSize(400,400);
}
Public static void main(String[]
                        args)
{
    new cardLayoutDemo();
}
```

DATE

Q.6) Explain two key swing feature.

→ two key's swing feature are:-

i) Swing components are lightweight

→ Swing is written in java and do not map directly to platform-specific peers. Lightweight components are more efficient and flexible.

→ Because lightweight components do not translate into native peers, the look and feel of each component is determined by swing, not by underlying OS. This means each component will work in a consistent manner across all platforms.

ii) Swing supports a pluggable look and feel

→ Because swing component is rendered by java code, look and feel of component is under control of swing.

→ It is possible to seperate look and feel of a component from logic of component and this is what Swing does.

→ It becomes possible to change the way that a component is re-

PAGE

ndered without affecting any of it's other aspects.
- In other words it is possible to "Plug in" a new look and feel for any given component without creating any side effects in code that uses that component.
- Advantages of pluggable look and feel :-
- It is possible to define look and feel that is consistent across all platform.
- possible to create look and feel that acts like a specific platform.
- It is also possible to design a custom look and feel.
- Java SE 6 provides metal and motif such look and feels. It is platform independent.

Q.7) Mention the key difference between swing and awt

->

DATE

AWT	Swing
→ Collection of GUI Components and other services for GUI programming in Java.	Part of Java foundation classes (JFC) used to create Java-based frontend GUI application
→ heavy weight components	lightweight components.
→ Platform dependent	Platform independent.
→ Doesn't support pluggable look and feel	Supports Pluggable look and feel.
→ Doesn't support MVC pattern	Supports MVC pattern.
→ Has to import java.awt	has to import javax.swing

Q.10) Difference Between SAX and DOM

→

→

DATE

SAX	DOM
-> It is simple API for XML parsing	It is called as Document object Model.
-> It is an event-based parser	It stays in tree structure
-> It is slower than DOM parser.	It is faster than SAX parser
-> It is best for smaller size of files	It is best for larger size of files
-> suitable for making XML files in java.	not good at making XML files in low memory.
-> It is read-only	It can insert or delete nodes.
-> backward navigation is not possible	backward and forward search is possible.
-> Small part of XML file is only loaded in memory.	Whole XML is loaded in memory.
-> Memory inefficient	Memory efficient.

PAGE

DATE

Q. 5) Explain UI delegate model.

=> UI delegate is the modified version of MVC (Model-View-Controller) that combines the view and controller into a single logical entity called UI delegate used by swing unlike classical implementation where (model-view-controller) as seperate entity.

-> However view (look) and controller (feel) are seperate from the model look and feel can be changed without affecting how component is used within a program.

-> UI delegates converesely is also possible (to customize model without affecting components

-> Eg. UI delegate for a button is Button UI.

-> UI delegates are classes that inherit ComponentUI

More:-

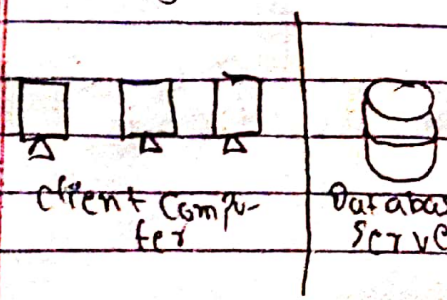
-> In swing API, each component class is are set named such as JButton, JTree, JTable, and so on. Each generic component class handles its own individual own view-and-controller responsibilities.

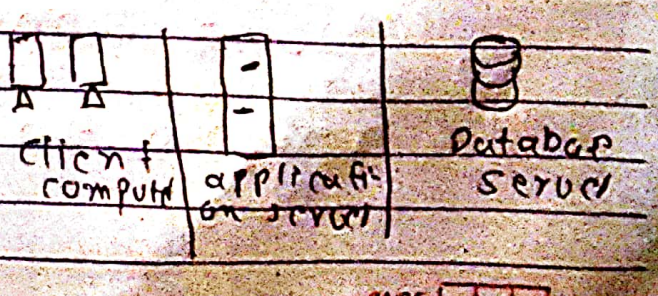
DATE

Each class delegates the look-and-feel-specific aspect of its responsibilities to whatever UI object the currently installed look-and-feel provides. For this reason, UI object that's built into every swing component is sometimes referred to as UI delegate.

* Differentiate between

2-tier	3-tier
-> Consists of client tier and database tier	Consists of clients application and database tier
-> Clients handle presentation and application layer and server system handles database layer	Client handles Presentation, application ~~handles~~ server handle application layer and Server system handles database layer
-> Good security	Exceptional security
-> Costly than 1-tier	Costly than 1 and 2-tier
-> No. Number of server ranges from 2-100	Number of server ranges 50-2000 (+)





PAGE

DATE

*)

→ Standalone application

Pros :-

a) application runs ~~cato~~ locally on device and doesn't need internet connection nor any other services installed.

b) needs no browser to run and is not bound to any specific platform.

Cons :-

a) User are restricted to specific computer.

b) The same software cannot be installed simultaneously i.e standalone requires that any new programs must be set up one by one, which is time consuming.

c) less secure

→ 2-tier application

Pros :-

a) This provides redundancy of network services.

b) limit acces to data to host.

Cons :-

a) Server cannot respond multiple request at same time

b) Can cause integrity issue as cannot respond to multiple request

PAGE

DATE

3 - tier application

Pros :-

i) Flexibility is increased as each tier can be managed seperately.
ii) Components of 3-tier are reusable
iii) more secure as data restriction.

Cons :-

i) High installation cost
ii) Structure is complex as compared to other tier.

PAGE