B.TECH.

THIRD SEMESTER EXAMINATION 2009-10

OBJECT ORIENTED SYSTEMS

Time : 3 Hours ***Total Marks : 100***

Note: Attempt all questions:

Note: Attempt any four parts: (5×4=20)

Q.1. (a) *What is object oriented methodology?*

Ans. The object oriented methodology is a methodology of software engineering, which is a process for the organized production of software, using a collection of predefined techniques and notational conventions. The steps of software production are usually organized into a life cycle (called software development life cycle) consisting of several phases as follows:

- Problem formation
- Analysis
- Design
- Implementation
- Testing
- Maintenance and
- Enhancement

Object Analysis: The purpose of analysis phase is to state and understanding the problem and modeling the application domain within which is operate. The analysis phase accepts input as a problem statement describing the problem to be solved along with a conceptual overview of the proposed system. The output from the analysis phase, is a formal model that captures the following aspects of the system:

1. Objects and their relationship (object model)
2. Dynamic flow of control (dynamic model)
3. Functional transformation of data subject to constraints (functional model)

Object Design: This phase determines the full definitions of the classes and associations used in the implementation, as well as the interfaces and algorithms of the methods used to implement operations. The objects design base adds internal objects for implementation and optimize data structure and algorithms. During the object design phase, the designer must perform the following steps:

(i) Combine the three models to obtain operations on classes.

(ii) Design algorithm to implement operations.

(iii) Optimize access paths to data.

(iv) Implement control for external interactions.

(v) Adjust class structure to increase inheritance.

(vi) Design associations.

(vii) Determine object representation.

(viii) Package classes and associations into modules.

Q.1. (b) Explain links and association with suitable example.

Ans. A link is a physical or conceptual connection between object instances. Mathematically a link is defined as a tuple, that is, an ordered list of object instances. A link is an instance of an association.

An association describes a group of links with common structure and common semantics. An association describes a set of potential links in the same way that a class describes a set of potential objects. Association are inherently bidirectional. In given figure shows one to one association and corresponding links.

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

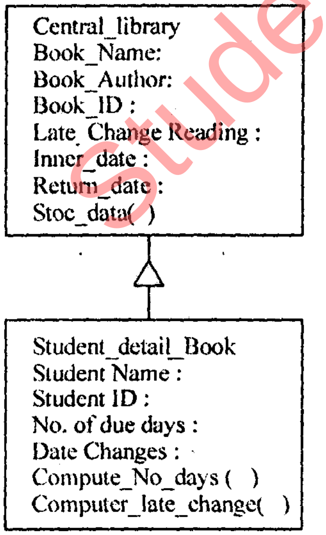
Each association in the class diagram corresponds to a set of links in the instance diagram. Each country has a capital city and capital is the name of association. The OMT notation for an association is a link between classes. A link is drawn as a line between objects.

Q.1. (c) Differentiate between aggregation and association.

Ans. In object-oriented programming, association defines a relationship between classes of object which allows one object instance to cause another to perform an action on its behalf. This relationship is structural, because it specifies that objects of one kind are connected to objects of another.

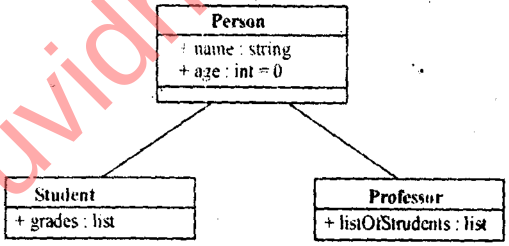
The aggregation is an extension of association means aggregation is a strong form of association in which an aggregate object is made of components. Components are part of aggregate. Thus aggregation is "part-whole" or "part-of" relationship e.g., Keyboard is the part of computer.

Q.1. (d) Prepare a portion of an object diagram for a library book checkout system that shows the data book in due and the late charges for an over due book as derived objects.



Q.1. (e) Explain generalizatioin and inheritance with example.

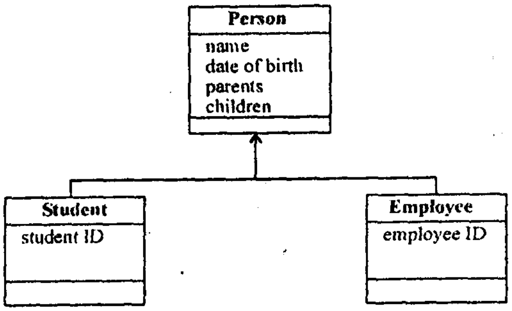
Ans. The generalization relationship indicates that one of the two related classes (the subtype) is considered to be a specialized form of the other (the supertype) and supertype is considered as 'Generalization' of subtype. The UML graphical representation of a Generalization is a hollow triangle shape of the supertype end of the line (or tree of lines) that connects it to one or more subtypes. The generalization relationship is also known as the inheritance or "is a" relationship. The supertype in the generalization relationship is also known as the "parent", superclass, base class, or base type. The subtype in the specialization relationship is also known as the "child", subclass, derived class, derived type, inheriting class, or inheriting type.



Inheritance in the object model is a means of defining one class in terms of another. Inheritance is a way to form new classes (instances of which are called objects) using classes that have already been defined. Inheritance is employed to help reuse existing code with little or no modification. The new classes, known as Sub-classes (or derived classes), inheirt attributes and behavior of the pre-existing classes, which aare referred to as Super-classes. In this example, a Student is a type of Person. Likewise, a Employee is a type of Person. Both Student and Employee inherit all the attributes and methods of Person. Student has a locally defined student ID attribute. Employee has a locally defined employee ID attribute.

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com



Q.1. (f) What is meta data? How it is important in object oriented methodology?

Ans. Meta is data that describe another data for exp the definition of a class is meta data. models are inherently meta data since they describe the things being modeled.

The term metadata encompasses all the information necessary to interpret, understand and use a given dataset. Discovery metadata more particularly apply to information (keywords) that can be used to identify and locate the data that meet the user's requirements (via a Web browser, a Web based catalogue, etc.). Detailed metadata include the additional information necessary for a user to work with the data without reference back to the data provider (although one element of the detailed metadata may be the data provider's contact!).

One major advantage of meta data is that redundancy and inconsistencies can be identified more easily as the meta data is centralised. For example the system catalogue and data dictionary can help or guide developers at the conceptual or structural phase or for further maintenance.

Metadata pertaining to observational data, for example, include details about how (with which instrument or technique), when, where and with which accuracy (or error bars) the data have been collected, by whom (including affiliation and contact address or telephone number) and in the framework of which research project. In the case of processed data, the nature of the initial raw data and the derivation process must be stated. The nature and units of the recorded variables are of course essential, as well as the grid or the reference system. Metadata pertaining to model output should include the name of the model, the conditions of the calculation, the type of constraint applied, the length of the integration, the nature of the output, the geographical domain over which the output is defined (when applicable), etc. Specific conditions applying to the model or the experiment may be mentioned. Metadata also obviously include information on the format in which the data are stored, the order of the variables, etc., to allow potential users to read them. Metadata pertaining to software models include the key points of the theory on which the model is based, the techniques and computational language used, references, etc.

Metadata relative to a specific data set can be provided as a separate document or as a piece of the data set itself. For digital data sets, this means that the metadata can sit in separate files (for example text files) or be integrated into the data file(s), as a header or at specified locations in the file. Some data formats provide room and rules for metadata.

Q.2. Attempt any four parts: (5×4=20)

(a) What is dynamic modeling? What is its importance?

Ans. Dynamic Model: The dynamic model describes those aspects of the system concerned with the sequencing of operations and time - events that cause state changes, sequences of events, states that define the context for events, and the organization of events and states. The dynamic model captures control information without regard for what the operations act on or how they are implemented.

The dynamic model includes event trace diagrams describing scenarios. An event is an external stimulus from one object to another, which occurs at a particular point in time. An event is a one-way transmission of information

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

from one object to another. A scenario is a sequence of events that occurs during one particular execution of a system. The dynamic model is represented graphically by state diagrams. A state corresponds to the interval between two events received by an object and describes the "value" of the object for that time period. A state is an abstraction of an object's attribute values and links, where sets of values are grouped together into a state according to properties that affect the general behaviour of the object. Each state diagram shows the state and event sequences permitted in a system for one object class. State diagrams also refer to other models: actions correspond to functions in the functional model; events correspond to operations on objects in the object model.

A dynamic model is constructed by following number of steps:

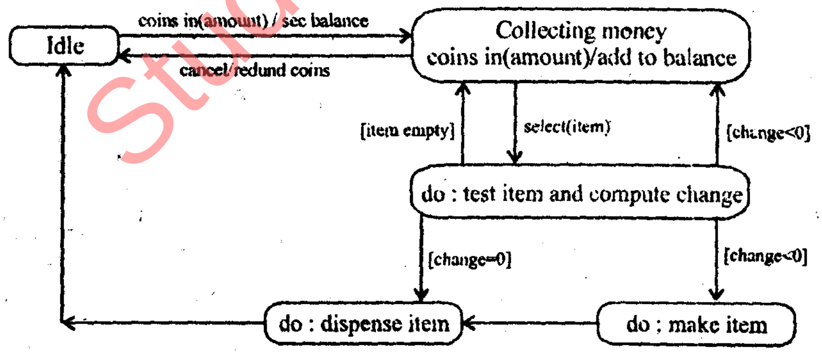
- Prepare scenarios of typical interaction sequences.
- Identify event between objects
- Prepare an event trace for each scenario.
- Build the state diagram.
- Match event between objects to verify consistency.

The advantage of dynamic model is that it describes the system behavior on occurrence of a particular event so that one can study the system behavior and can take specific steps to improve the operations of the system. Controlling the system after observing the dynamic behavior is easy.

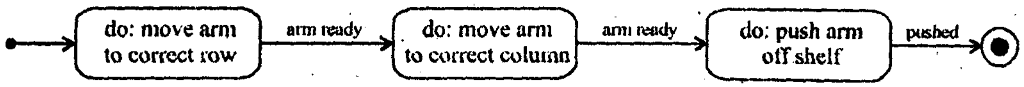
Q.2. (b) What are nested state diagrams? Explain by taking a suitable example.

Ans. Nested State Diagram: State diagrsms can be structured to permit concise description of complex system. The ways of structuring state machine are similar to the ways of structuring objects: generalization and aggregation. Generalization is equivalent to expanding nested activities. It allows an activity to be described at a high level, then expanded at a lower level by adding details, similar to procedural call. For example shows the type level model for a vending machine. This diagram contains an activity dispense item and an event select(item) that are expanded in more detail in nested state Diagram. The event coins in(amount) is written within collecting money state. This indicates a transition that remains within a single state. Also, the transition from the unnamed state containing "do: dispense item" to state Idle has no event label. The lack of event label indicates that the transition fires automatically when the activity in the state is complete.

StudentSuvidha.com



Vending machine model



Dispense item activity of vending machine

Download All Btech Stuff From StudentSuvidha.com

(c) Prepare a list of objects that you would expect of the system to handle a Telephone Answering Machine.

Ans. Telephone answering machines perform the basic function that they automatically answer calls incoming on a telephone line if it is not answered first, play a prerecorded outgoing message to the calling party, and then record any message the calling party leaves. A typical telephone answering machine is a stand-alone device coupled to a telephone line. This type of answering machine typically includes circuitry for detecting a ringing signal on the telephone line indicating the presence of an incoming telephone call, answering the call by taking the line off-hook, playing an outgoing or "greeting" message, recording an incoming message and hanging up the line in order to respond to a subsequent telephone call. The greeting and incoming recorded messages typically are recorded in the analog domain using one or more conventional audio tapes. The machine may have a display for indicating the number of messages received and the time and date they were received. A telephone with an answering function or mode which automatically records the message of a caller on a recording medium like a cassette tape or an endless tape when nobody is available to answer the telephone. Telephones of this type have an answer mode switch. When this switch is on, the phone mode is changed to the answering mode when receiving a call, and when the switch is off, the phone mode is set to a normal phone mode. Telephone answering machines may also include circuitry for enabling remote-control operation of various ones of the machine's functions (e.g., playback of recorded messages, fast forwarding and rewinding of recorded messages, recording a new greeting message, and so forth). The owner may call his or her home telephone number, wait for the answering machine to answer the call, and then enter a DTMF access code. If the answering machine recognizes the access code, it will allow the user to retrieve messages, delete messages, turn the machine on or off, and perform other control functions.

List of objects:

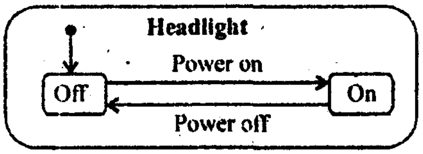
- Microprocessor
- LCD OR LED Display
- Recording and playback device
- Indicator lights
- Audio Controller
- Telephone Line
- Receiver
- Caller

(d) Explain event and state by taking a suitable example.

Ans. Event: An event is something that happens at a point in time, such as user depresses left button or Flight 123 departs from Chicago. An event has no duration. Two events that are causally unrelated are said to be concurrent; they have no effect on each other. An event is one way transmission of information from one object to another. An event convey information from one object to another. The data values conveyed by an event are its attributes like the data values held by the objects. Attributes are shown in parentheses after the event class name.

State: A state is an abstraction of the attribute values and links of an object. A state specifies the response of objects to input events for example. If a digit is dialed in state Dial tone, the phone line drops the dial tone and enters state dialing; if the receiver is replaced in state dial tone, the phone line goes dead and enters state Idle. A state has no duration; it occupies an interval of time. A state is associated with the value of an object satisfying some.

In the above figure there are two states on and off and the events are power on and power off.



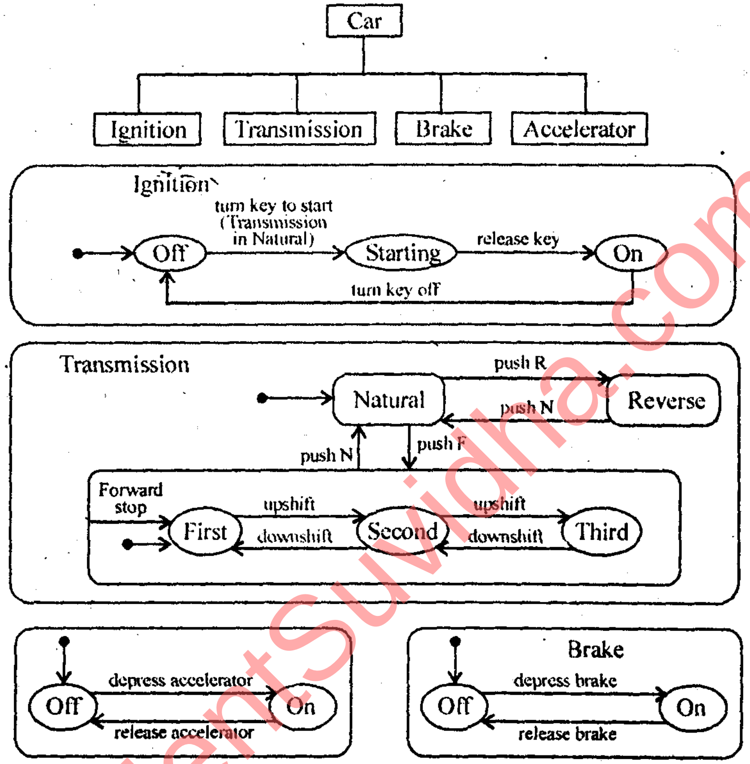
(e) What do you mean by aggregation concurrency?

Ans. Aggregation implies concurrency. The aggregate state corresponds to the combined state of all components in diagram. Aggregation is said to be the "and relationship". The aggregate state is one state form first diagram and a state from second diagram and a state

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

from each other diagram. Guarded transition for one object can depend on another object being in a given state. This allows interaction between state diagrams, while preserving modularity. For eample, given figure shows the state of car as an aggregation of component states: the ignition, Transmission, acceleration and brake. Each component also has substate. The state of car includes one substate for each component.



StudentSuvidha.com

In the example ignition, transmission, break and accelerator all are the part of the car. All of these are collectively and concurrently work together to perform the integrated function of the whole car.

(f) Discuss various ways of implementing DFD.

Ans. Ways to implement data-flow diagram: It is common practice to draw a context-level data flow diagram first, which shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modelled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization.

Top-down approach:

1. The system designer makes "a context level DFD" or level 0, which shows the "interaction" (data flows) between "the system" (represented by one process) and "the system environment" (represented by terminators).
2. The system is "decomposed in lower-level DFD (Level 1)" into a set of "processes, data stores, and the data flows between these processes and data stores."
3. Each process is then decomposed into an "even-lower-level diagram containing its subprocesses".
4. This approach "then continues on the subsequent subprocesses", until a necessary and

Download All Btech Stuff From StudentSuvidha.com

sufficient level of detail is reached which is called the primitive process. DFD is also a virtually designable diagram that technically or diagrammatically describes the inflow and outflow of data or information that is provided by the external entity.

- In level 0 the diagram does not contain any DataStores.

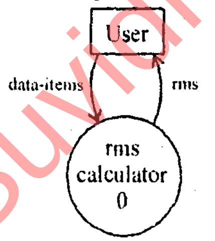
Event partitioning approach: A context level data flow diagram created using Select SSADM.

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are "owned" by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entites.

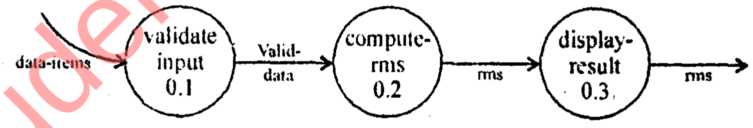
Level 1 (high level diagram): A Level 1 Data flow diagram for the same system.

This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major high-level processes of the system and their interrelation. A process model will have one and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down to more detail in the level 1, e.g. the "inquiry" data flow could be split into "inquiry request" and "inquiry results" and still be valid.

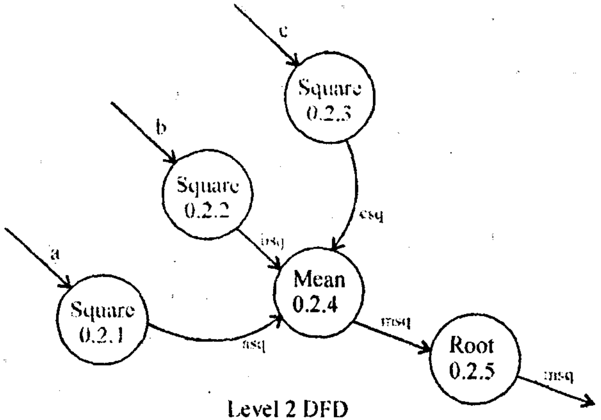
Level 2 Diagram: In the level 2 diagram we can further expand the level 1 diagram and the numbering in the process shows the level of the DFD. A level 1 DFD has process numbers as, 0.1, 0.2, 0.3, ... whereas a level 2 DFD has process numbers as 0.2.1, 0.2.2, 0.2.3, ...



Context diagram



Level 1 DFD



Level 2 DFD

Context diagram, level 1 and level 2 DFDs.

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

Q.3. Attempt any two parts: **(2×10=20)**

(a) What is functional modeling? What are its advantages? Describe the relation of functional model, object model and dynamic model.

Ans. The functional model describes computations within a system. The functional model shows how output values in a computation are derived from input values, without regard for the order in which the values are computed. Functional model is represented by data flow diagram. A data flow diagram consists of processes, data stores, actors and data flow.

Advantages of functional modeling:

- It shows explicitly the data flow among the processes.
- It reduces the complexity of the system by dividing it into processes.

Relation with Object and Dynamic model:

- The object model shows the static structure of the real world, the dynamic model shows the behavior of the system over time and the functional model shows the functional derivation of values, regardless for when they are computed.
- The functional model specifies what happens, the dynamic model specifies when it happens, and the objct model specifies what it happens to.

(b) What is JSD? Compare between SA/SD and OMT modeling.

Ans. Jackson Structured Development (JSD) is another methodology of software engineering. JSD has a different development style than SA/SD or OMT. JSD does not distinguish between analysis and design and instead emphasizes on both phases together as specification. JSD divides system development into two stages: specification, then implementing. A JSD model describes the real-world in terms of entities, actions and ordering of actions. JSD software development consists of six sequential steps: (i) Entity action, (ii) Entity structure, (iii) Initial model, (iv) Function, (v) System timing, (vi) Implementation.

Comparison of SA/SD and OMT Approach:

- In SA/SD approach, the functional model dominates, the dynamic model is next important and object model is least important. In contrast object modeling regards object model is as most important then dynamic model and finally functional model.
- SA/SD approach organize system around procedures. In contrast, OMT organize system around procedures.
- An object oriented design is more resilient to change and are more extensible whereas SA/SD approach has well defined system boundaries across which system must communicate with the real world.
- In SA/SD approach, decomposition of process into sub process is somewhat arbitrary whereas in OMT, the decomposition is based on objects in problem domain.
- An object oriented approach better integrates database with programming code whereas in SA/SD approach it is difficult to integrate database with programming code.

(c) What is Java foundation classes? Prepare a data flow diagram for computing the volume and a surface of cylinder. Inputs are the height and radius of the cylinder. Output are volume and surface area. Discuss several way of implementing the data flow diagram.

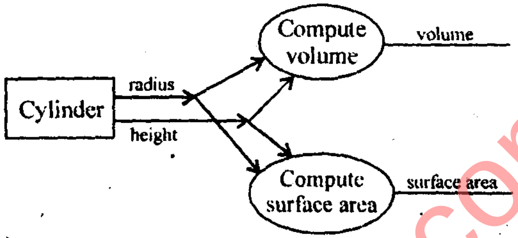
Ans. The Java Foundation Classes (JFC) are a graphical framework for building portable

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

Java-based graphical user interfaces (GUIs). JFC consists of the Abstract Window Toolkit (AWT), Swing and Java 2D. Together, they provide a consistent user interface for Java programs, regardless whether the underlying user interface system is Windows, Mac OS X or Linux.

Using the Java programming language, Java Foundation Classes (JFC) are pre-written code in the form of class libraries (coded routines) that give the programmer a comprehensive set of graphical user interface (GUI) routines to use.



Ways to implement data-flow diagram: It is common practice to draw a context-level data flow diagram first, which shows the interaction between the system and external agents which act as data sources and data sinks. On the context diagram (also known as the Level 0 DFD) the system's interactions with the outside world are modelled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organizaton.

Top-down approach:

1. The system designer makes "a context level DFD" or Level 0, which shows the "interaction" (data flows) between "the system" (represented by one process) and "[illegible] environment" (represented by terminators).

2. The system is "decomposed in lower-le[illegible] ocesses, data stores, and the data flows between these processes [illegible]

3. Each process is then decomposed into an "even-lower-level diagram containing its subprocesses."

4. This approach "then continues on the subsequent subprocesses", until a necessary and sufficient level of detail is reached which is called the primitive process.

DFD is also a virtually designable diagram that technically or diagrammatically describes the inflow and outflow of data or information that is provided by the external entity.

In Level 0 the diagram does not contain any DataStores.

Even partitioning approach: A context level Data flow diagram created using Select SSADM.

This level shows the overall context of the system and its operating environment and shows the whole system as just one process. It does not usually show data stores, unless they are "owned" by external systems, e.g. are accessed by but not maintained by this system, however, these are often shown as external entities.

Level 1 (high level diagram): A Level 1 Data flow diagram for the same system.

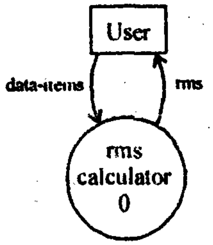
This level (level 1) shows all processes at the first level of numbering, data stores, external entities and the data flows between them. The purpose of this level is to show the major high-level processes of the system and their interrelation. A process model will have one, and only one, level-1 diagram. A level-1 diagram must be balanced with its parent context level diagram, i.e. there must be the same external entities and the same data flows, these can be broken down

StudentSuvidha.com

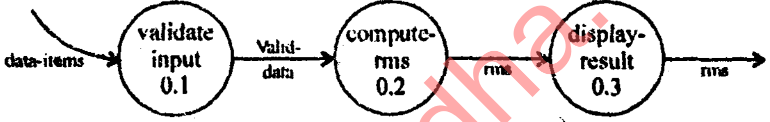
Download All Btech Stuff From StudentSuvidha.com

to more detail in the level 1, e.g. the "inquiry" data flow could be split into "inquiry request" and "inquiry results" and still be valid.

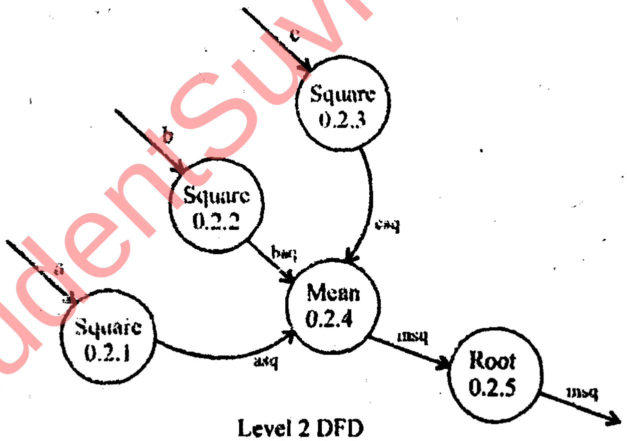
Level 2 Diagram: In the level 2 diagram we can further expand the level 1 diagram and the numbering in the process shows the level of the DFD. A level DFD has process numbers as, 0.1, 0.2, 0.3.... whereas a level 2 DFD has process numbers as 0.2.1, 0.2.2, 0.2.3, ...



Context diagram



Level 1 DFD



Level 2 DFD

StudentSuvidha.com

Context diagram, level 1 and level 2 DFDs.

Q.4. Attempt any four parts: **(5×4=20)**

(a) How are methods and data organised in an object oriented program? With an example.

Ans. Data and methods are encapsulated in a single unit called class data and methods can be either public private protected or default in a class depends on the scope.

Data are declared as attributes at the top of the class body and methods are defined below the data section.

Example:

Class myclass

{

Download All Btech Stuff From StudentSuvidha.com

```
Int a, b, sum;
myclass( )
{
a=6;
b=7;
}
public int add( )
{
sum=a+b;
return sum;
}
public static void main(String arg[ ])
{
myclass mc=new myclass( );
System.out.println("sum="+mc.sum( ));
}}
```

In the example data are a, b and sum. The methods are add( ) and a main method.

(b) What are command line arguments? Write a program for finding factorial for a given number using command line argument.

Ans. A Java application can accept any number of arguments from the command line. This allows the user to specify configuration information when the application is launched.

The user enters command-line arguments when invoking the application and specifies them after the name of the class to be run. For example, suppose a Java application called Sort sorts lines in a file. To sort the data in a file named friends.txt, a user would enter:

```
java Sort friends.txt
```

When an application is launched, the runtime system passes the command-line arguments to the application's main method via an array of Strings.

```
public class findfact
{
int f, resfact, i;
void computefact( )
{
f=integer.parseint(args[0]);
i=f-1;
while (i>=1)
{
f=f*i;
i–
}}
public static void main(String arg[ ])
```

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

```
{
findfact fd=new findfact( );
fd.computefact( );
}}
```

Q.4. (c) What is type casting? What is it required in programming?

Ans. Type Casting refers to changing an entity of one datatype into another. This is important for the type conversion in developing any application. If you will store a int value into a byte variable directly, this will be illegal operation.

It is of two types type promotion and type demotion, when we assign a higher value in lower variable it is called type demotion and when we assign a lower value in a higher variable it is called type promotion.

For storing your calculated int value in a byte variable you will have to change the type of resultant data which has to be stored.

Exp:

```
Type1 var1=value1;
Type2 var2=value2
Var1=(Type1)var2
```

Q.4. (d) What is copy constructor? Explain with example.

Ans. A copy constructor is a constructor that takes only one parameter that is the same type as the class in which the copy constructor is defined. Suppose we define a class called X. Then, a copy constructor defined in class X would also expect a parameter of type X.

A copy constructor is used to create an object that's a copy of its parameter, but is actually an independent copy. It's independent in the sense that the copy is located at a completely different address in memory than the original.

```
public copycon
{
int a,b,sum;
copycon( ){a=6;
b=7;
}
public int add( )
{
sum=a+b;
return sum;
}
public static void main(String arg[ ])
{
copycon cc;
copycon mc=new copycon( );
cc=mc;
```

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

```
System.out.println("sum="+mc.sum( ));
System.out.println("sum="+cc.sum( ));
}}
```

Q.4. (e) What is an interface? Write a program to find the area of the circle by using interface.

Ans. In the Java programming language, an interface is a reference type, similar to a class, that can contain only constants, method signatures, and nested types. There are no method bodies. Interfaces cannot be instantiated—they can only be implemented by classes or extended by other interfaces.

```
interface computearea
{
public final double pi=3.14;
public doube cir_area( );
}
class circle implements computearea
{
double radius, area;
circle( )
{
radius=4.5;
}
public double cir_area( )
{
area=pi*radius*radius;
return area;
}
public static void main(String arg[ ])
{
computearea ca=new circle( );
System.out.println("area="+ca.cir_area( ));
}}
```

StudentSuvidha.com

Q.4. (f) How many types of layout manager are used in java programming? Describe each type in short.

Ans. Layout Manager: Java technology uses Layout Managers to define the location and size of Graphical User Interface components. It does not encourage programmers to use absolute location and size of components. Java technology instead places components based on specified Layout Manager. To add components to a Frame instance, you need to specify the type of component layout scheme to be used. It is used to specify to a container how components within that container should be arranged. Types of Layout Manager are:

1. FlowLayout: The FlowLayout manager is the simplest of the layout managers. In this scheme, components are arranged in the order they are placed within the container. If a row becomes completely filled, the remaining components are grouped together on the next row. An example of a FlowLayout, using button components,

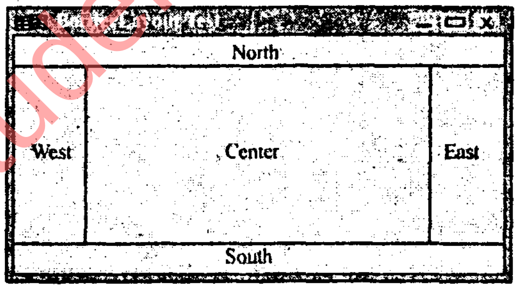
```
import java.awat.*;
public class FlowLayoutExample extends Frame {
      public FlowLayoutExample(String title, int count){
            super(title);
                  setLayout(new FlowLayout( ));
                  for(int i=1; i<=count; i++)
                        add(new Button(Integer.toString(i)));
}}
```

2. BorderLayout: The BorderLayout manager arranges components within specified regions of a container. Valid regions are “North”, “South”, “East”, “West” and “Center”. An example of the BorderLayout layout manager.

The code for BorderLayoutExample is shown below. Since the default layout manager for frames is BorderLayout, we do not explicitly set the layout manager.

```
public class BorderLayoutExample extends Frame {
      public BorderLayoutExample(String title) {
            super(title);
                  add(“North”, new Button(“North”));
                  add(“South”, new Button(“South”));
                  add(“East”, new Button(“East”));
                  add(“West”, new Button(“West”));;
                  add(“Center”, new Button(“Center”))
}}
```

StudentSuvidha.com



3. GridLayout: A container using the GridLayout scheme arranges components in rows and columns in row-major order. Each component is sized to fit its respective grid cell. An example of a GridLayout is shown below:

```
import java.awat.*;
public class GridLayoutExample extneds Frame {
      public GridLayoutExample(String title, int rows, int columns) {
            super(title);
                  setLayout(new GridLayout(rows, columns));
```

Download All Btech Stuff From StudentSuvidha.com

```
            for(int i = 0; i<rows; i++)
                for(int j=0; j<columns; j++)
                    add(new Button(Integer.toString(i)+ "," + Integer.toString(j)));
    }
}
```

4. Card Layout: It is a layout manager for a container. It treats each component in the container as a card. Only one card is visible at a time, and the container acts as a stack of cards. The first component added to a CardLayout object is the visible component when the container is first displayed. The ordering of cards is determined by the container's own internal ordering of its component objects. CardLayout defines a set of methods that allow an application to flip through these cards sequentially, or to show a specified card.

```
import java.awat.*;
import java.applet.Applet;
import java.awt.event.*;
public class Card extends Applet implements ActionListener
{
    Panel cardPanel;
    Panel firstP;
    Panel buttonP;
    Button first;
    CardLayout ourLayout;
    public void init( )
    {
        cardPanel = new Panel( );
        ourLayout = new CardLayout( );
        cardPanel.setLayout (ourLayout);
        firstP = new Panel( );
    firstP.setBackground(Color.blue);
    first = new Button("First");
    first.addActionListener(this);
    buttonP = new Panel( );  //Panel's default Layout manager is FlowLayout
    buttonP.add(first);
    r
    this.add(buttonP, BorderLayout.SOUTH);
    this.add(cardPanel, BorderLayout.CENTER);
    cardPanel.add(firstP, "First");
    }}
```

Q.5. Attempt any two parts: (2×10=20)

(a) What is swing? How it differs with AWT? Write a program in swing which shows two input boxes names uses id and passwords.

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

Ans. Swing is used to create a Java program with a graphical user interface (GUI). The Swing toolkit includes a rich set of components for building GUIs and adding interactivity to Java applications. It includes all the components from a modern toolkit: table controls, list controls, tree controls, buttons, and labels. It is part of the Java Foundation Classes (JFC). The JFC also include other features important to a GUI program, such as the ability to add rich graphics functionality and the ability to create a program that can work in different languages and by users with different input devices.

Comparison of AWT and Swing:

- Swing is a set of classes that provides more powerful and flexible comonents than are possible with the AWT.
- In additiion to the familiar components, such as buttons, check boxes, and labels, Swing supplies several exciting additions, including tabbed panes, scroll panes, trees, and tables.
- Even familiar components such as buttons have more capabilities in Swing. For example, a button may have both an image and a text string associated with it. Also, the image can be changed as the state of the button changes.
- Unlike AWT components, Swing components are not implemented by platform-specific code. Instead, they are writtn entirely in Java and, therefore, are platform-independent. The term lightweight is used to describe such elements.

```
import javax.swing.*;
Class mylogin extends JFrame
{
JLabel ln, lp;
JTextField jt, jn;
mylogin( )
{
ln=new JLabel(“userid”);
tn=new JTextField(“ ”);
lp=new JLabel(“userid”);
tp=new JTextField(“userid”);
setLayout(new FlowLayout( ));
add(ln);
add(tn);
add(lp);
add(tp);
setVisible(true);
}
public static void main(String arg[ ])
{
Mylongin ml=new mylogin( );
}}
```

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

Q.5. (b) What is applet, define its life cycle? How servlets are different from applet? Write a program in servlet which prints the word "hello".

Ans. An applet is a program written in the Java programming language that can be included in an HTML page, much in the same way an image is included in a page. When you use a Java technology-enabled browser to view a page that contains an applet, the applet's code is transferred to your system and executed by the browser's Java Virtual Machine (JVM).

Applet runs in the browser and its lifecycle method are called by JVM when it is loaded and destroyed. Here is the lifecycle methods of an Applet:

int( ): This method is called to initialize an applet.

start( ): This method is called after the initialization of the applet.

stop( ): This method can be called multiple times in the life cycle of an Applet.

destroy( ): This method is called only once in the life cycle of the applet when applet is destroyed.

Difference between Sevlet and Applet:

- Applet is a part of Core JAVA and Servlet of Advance Java. Applet is client side program and Servlet is Server side. When Applet runs it take the resources of client whereas Servlet is processed at server.
- An Applet's class, jar files can be accessed and downloadable by client but not so in case of servlet.
- Applets can run under any web browser their execution is dependent on Client as they require JRE. Whereas Servlets do not require any thing specific at client side, as they require java enabled web/application Server.
- Servlets are to servers what applets are to browsers.
- Applets must have graphical user interfaces where as servlets have no graphical user interfaces.

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class HelloWorld extends HttpServlet{
public void doGet (HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException{
        response.setContentType("text/html");
        PrintWriter pw = response.getWriter( );
        pw.println("<html>");
        pw.println("<head><title>Hellow</title></title>");
        pw.println("<body>");
        pw.println("<h1>hellow</h1>");
        pw.println("</body><html>");
    }
}
```

Q.5. (c) Write short notes on any two:

(i) JDBC

StudentSuvidha.com

(ii) Difference between Java and C++

(iii) Java Beans

(iv) Lavatron applet

Ans. (*i*) JDBC: Java Database Connectivity or in short JDBC is a technology that enables the java program to manipulate data stored into the database. JDBC is Java application programming interface that allows the Java programmers to access database management system from Java code. It was developed by JavaSoft, subsidiary of Sum Microsystems.

JDBC (the Java Database Connection) is the standard method of accessing databases from Java. Sun developed the JDBC library after considering Microsoft's ODBC. Their aims were to get something similar but easier to learn and use: ODBC is complex because it has a few very complex calls. JDBC has split up this complexity into many more calls, but with each of them being relatively simple.

Accessing a database using JDBC involves a number of steps:

1. Get a Connection object connected to a database.
2. Get a Statement object from an open Connection object.
3. Get a ResultSet from a Statement's query execution.
4. Process the rows from the ResultSet.

Example:

```
import java.sql.*;
public class jdbc1
{     public static void main(String args[ ])
      {
           ResultSet r;
           try
           {
           Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
           Connection c=DriverManager.getConnection("jdbc:odbc:stud_dns");
           Statement st=c.createStatement( );
           r=st.executeQuery("select * from stud");
           System.out.println("Reg no\tName\tClass\tMark");
           System.out.println("**********************");
           while(r.next( ))
System.out.println(r.getInt(1)+ "\t"+r.getString(2)+ "\t"+r.getString(3)+ "\t"+r.getInt
           }
           catch(SQLException e)
           {
           System.out.println("SQL ERROR:"+e);
           }
           catch(Exception e)
           {
```

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

```
System.out.println("ERROR:"+E);
}}}
```

(ii) Difference between Java and C++: There are also significant differences from C++, which fundamentally makes Java distinct from C++. Perhaps the single biggest difference between Java and C++ is that Java does not support pointers. Pointers are inherently insecure and troublesome. Since pointers do not exist in Java, neither does the → operator. Some other C++ features are not found in Java.

- Java does not include structures or unions because the class encompasses these other forms. It is redundant to include them.
- Java does not support operator overloading.
- Java does not include a preprocessor or support the preprocessor directives.
- Java does not perform any automatic type conversions that result in a loss of precision.
- All the code in a Java program is encapsulated within one or more classes. Therefore, Java does ot have global variables or global functions.
- Java does not support multiple inheritance.
- Java does not support destructors, but rather, add the finalize( ) function.
- Java does not have the delete operator.
- The <<and>> are not overloaded for I/O operations.
- Java does not support templates.

(iii) Java Beans: A Java Bean is a reusable software component that can be visually manipulated in builder tools. Individual Java Beans will vary in functionality, but most share certain common defining features:

- Support for introspection allowing a builder tool to analyze how a bean works.
- Support for customizatio allowing a user to alter the appearance and behavior of a bean.
- Support for events allowing bean to fire events, and informing builder tools about both the events they can fire and the events they can handle.
- Support for properties allowing beans to be manipulated programmatically, as well as to support the customization mentioned above.
- Support for persistence allowing beans that have been customized in an application builder to have their state saved and restored. Typically persistence is used with an application builder's save and loan menu commands to restore any work that has gone into constructing an application.

Getters and Setters: Variables in a JavaBean normally have two methods associated with each variable, a get method and a set method. The get method, or getter, retrieves the value stored in the variable. The set method, or setter, sets the value for the variable. Though it is normal to have a get and a set method for each variable, it is not required. For example, a read only bean that has getters, but o setters. Also, a write only bean with setters, but no getters. Their are uses for either. However, commonly a bean has both as in our example. Here's the bean with its getters and setters.

Example: SavlesTaxBean.java

```
public class SalesTaxBean{
//Two Attributes
private double taxRate;
private double amount;

public SalesTaxBean( ){ //Sample zero argument constructor
```

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com

```
setTaxRate(“0”);
setAmount(“0”);
}

public double getTaxRate( ){ //Tax Rate Gettr
return taxRate;
}

public void setTaxRate(String new TaxRate){ //Tax Rate Setter
taxRate = Double . valueOf(newTaxRate).doubleValue( )/100;
}
```

(iv) Lavatron applet: Lavatron is a sports arena lightbulb display. David LaVallee, the author of the ImageMenu applet, wanted to achieve this kind of effect for a long time. The history of Lavatron begins way back in 1974, when LaVallee was the stick boy for the California Golden Seals of the NHL. David recalls, “Out scoreboard just displayed, well, the score. In 1979, LaVallee became fascinated with the idea of a graphical programmable scoreboard when he was the repair guy for the Digital Equipment Corporation PDP 11/34 that ran the scoreboard at the Canadian National Exhibition Stadium (where the Toronto Blue Jays used to play). That scoreboard was based on plain old 100-watt lightbulbs like you use at home. In 1995, Lavatron was written again from scratch to run under Java, and it has undergone several performance tweaks and iterations since.”

How Lavatron Works: Lavatron is able to present an interesting image onscreen because of a small trick that it employs, and its side effect allows the applet to load very quickly. The reason it loads so quickly is that there isn’t much data transmitted over the Net. The source image is a JPEG image that is 64 times smaller than the displayed image. Each pixel in the source image is scaled up to an 8×8-pixel image of a transparent circle surrounded by a black bezel, with a white highlight for a dash of style, is painted over the scaled-up color pixel. As an optimization, the bulbs are preassembled into an image that can be painted once for each column.

Lavatron paints so fast becaue it doesn’t have to repaint what it has already drawn. The technique of copying the area of the screen that’s good and painting just the portion that’s new is used in many common operations involving scrolling. The awt.Graphics unction copyArea( ) takes a portion of an image defined by a rectangle and moves it by an x, y offset from its starting location. As a graphics speed optimization, copyArea( ) is hard to beat. It such as the use of drawImage( ), or drawImage( ) through a clipRect( ).

The Source Code

The APPLET Tag: The source code starts with an APPLET tag for Lavatron, shown here. This applet looks best when the width is an even multiple of the bulb size and the height is the bulb size times the source image height. The only parameter is for the name of the source image file, named in img.

```
<applet code=Lavatron.class width=560 height=128>
<param name= “img” value=“swsm.jpg”>
</applet>
```

StudentSuvidha.com

Download All Btech Stuff From StudentSuvidha.com