Subject–Santali, Semester–IV, Paper–SNT-404, Santali Literary Essays

For–ILMS

By – Dr. Ratan Hembram, Asst. Prof., Dept. of Santali, V.U.

"[illegible]"

[illegible]

"[illegible]"

1. [illegible]

2. [illegible]

29-12-08

PAGE NO.:

1. Introduction
2. Concept of Inheritance
3. Concept of Polymorphism
4. Concept of Abstract Class & Interface
5. Conclusion

Inheritance is the process by which one object acquires the properties of another object. With the use of inheritance the information is made manageable in a hierarchical order. When we talk about inheritance, the most commonly used example is Animal Kingdom, where the Animal class is the base class in the hierarchy, and the inheritance relationship of Human, Apes, Monkey, Tiger, Cat, Dog, Horse, Camel, Pig, Rabbit, Bat, Lion, Bear, Deer, Elephant, Leopard, Giraffe, Fox, Wolf, Chimpanzee, Snake, Alligator, Crocodile, Turtle, Fish, Octopus, Starfish, Coral, Insect, Spider, Scorpion, Snail, Slug, Worm, Jellyfish, Flatworm, Sponge, Amoeba and Paramecium. This is very intuitive when you want to model relationships. Inheritance is also known as "is-a" relationship. For example, Human is-a Animal, Dog is-a Animal, Apes is-a Animal, Monkey is-a Animal, Human is-a Apes, Monkey is-a Apes etc. Inheritance is used to model is-a relationship. Don't use inheritance just to get code reuse. If is-a relationship is not satisfied, don't use inheritance even though you get code reuse. Otherwise it leads to problems later. Inheritance is used to indicate 'is-a-like-a' relationship. This reason should be used very sparingly and only when it is absolutely required. One use of inheritance is to achieve polymorphism in object oriented systems. In OOPs, we can call upon an object to perform an action, when the object is not exactly of the type that is expected but one of its descendants. The benefit of this is that code does not have to be rewritten if a descendant object is used for the first time. Polymorphism is the ability of objects belonging to different types to respond to method, field or property calls of the same name, each one according to an appropriate type-specific behavior. The programmer (and the program) does not have to know the exact type of the object in advance, and so the exact behavior is determined at run time (this is called late binding or dynamic binding). Some authors distinguish polymorphism from duck typing in that subtype polymorphism or inclusion polymorphism is usually limited by explicit or implicit relationships, whereas duck typing allows any objects that happen to provide the appropriate methods, fields or properties. Abstract classes are classes that contain one or more abstract methods, which are methods that are declared, but contain no implementation. Abstract classes may not be instantiated, and require subclasses to provide implementations for the abstract methods. Subclasses of abstract classes must implement all abstract methods in order to be instantiated. Abstract classes are useful when creating designs because they allow a developer to specify the methods that a class must implement, without having to specify how they must be implemented. Abstract classes are also useful when creating designs because they allow a developer to provide some default implementation of a method, knowing that it will be overridden if necessary by more specific classes. Interfaces are a syntactical contract that should be followed by the classes that implement them. Interfaces define the syntax of the contract and leave the semantics (i.e. the behavior) up to the implementing classes. Interfaces provide a mechanism for specifying a relationship between otherwise unrelated classes, typically by specifying a set of common methods each implementing class must contain. Interfaces are useful for hiding implementation details. They provide a clean separation between what a class does and how it does it, allowing the programmer to program "to the interface". Interfaces are useful for separating responsibilities. Developers can implement two unrelated interfaces in one class, and provide clients two narrow views of that class, each view containing only the methods relevant to that client. Interfaces are useful for modeling multiple inheritance. Multiple inheritance is a feature of some object oriented computer languages in which a class can inherit behaviors and features from more than one superclass. Such a class combines (inherits) the behaviors of all of the classes it extends. Multiple inheritance is controversial because it can be difficult to predict which inherited behavior will be invoked when calling upon an object that is an instance of a class that inherits from more than one superclass. Interfaces are useful for modeling multiple inheritance. Multiple inheritance is a feature of some object oriented computer languages in which a class can inherit behaviors and features from more than one superclass. Such a class combines (inherits) the behaviors of all of the classes it extends. Multiple inheritance is controversial because it can be difficult to predict which inherited behavior will be invoked when calling upon an object that is an instance of a class that inherits from more than one superclass.

Teacher's Signature

[illegible]

1. [illegible]
2. [illegible]
3. [illegible]
4. [illegible]
5. [illegible]

[illegible]

PAGE NO.:
DATE:

3. Explain the concept of Data Warehouse. Explain the three-tier architecture of Data Warehouse.

A Data Warehouse is a subject-oriented, integrated, time-variant, and non-volatile collection of data that supports management's decision-making process. It is designed to provide a consistent and comprehensive view of an organization's data, which is typically derived from multiple operational systems. The data in a Data Warehouse is usually historical and is used for analysis and reporting rather than for day-to-day transactions.

The three-tier architecture of a Data Warehouse consists of the following layers:

1. Bottom Tier: This is the data warehouse database server. It is where the data is stored and managed. The data is usually extracted from various operational systems, transformed into a consistent format, and loaded into the data warehouse using ETL (Extract, Transform, Load) tools.

2. Middle Tier: This is the OLAP (Online Analytical Processing) server. It provides a multidimensional view of the data and supports complex analytical queries. The OLAP server can be implemented using either a relational database (ROLAP) or a multidimensional database (MOLAP).

3. Top Tier: This is the front-end client layer. It includes tools for querying, reporting, analysis, and data mining. Users can interact with the data warehouse using these tools to generate reports, perform ad-hoc queries, and visualize data.

The three-tier architecture provides a separation of concerns between the data storage, processing, and presentation layers, making it easier to maintain and scale the data warehouse system.

Teacher's Signature

31-12-08

4— [illegible]

[illegible] 1750 [illegible]

[illegible] 1855 [illegible]

[illegible]

[illegible]
[illegible]
[illegible]
[illegible]

5. [illegible]
[illegible]
[illegible]
[illegible]
[illegible]
[illegible]

[illegible]

31-7-12

[illegible]

[illegible]

[illegible] 19

[illegible]

[illegible]

[illegible]

[illegible]

17-7-14

[illegible]

[illegible] God helps

[illegible] English [illegible]

3- Kindness

"भगवान उसी मनुष्य की सहायता करता है जो अपनी सहायता वह स्वयं
करता है।" Heaven helps those who help themselves.

2- Hard work leads to success.

3- Honesty is the best policy.

4- Unity is strength.

5- Time and tide wait for none.

6- Where there is a will there is a way.

7- Knowledge is power.

8- Health is wealth.

9- Slow and steady wins the race.

10- Practice makes a man perfect.

11- Necessity is the mother of invention.

12- Speech is silver but silence is gold.

13- Cleanliness is next to godliness.

14- Charity begins at home.

15- Look before you leap.

16- Better late than never.

17- Birds of a feather flock together.

18- Actions speak louder than words.

19- Every cloud has a silver lining.

20- Rome was not built in a day.

21- Barking dogs seldom bite.

22- Nothing ventured nothing gained.

23- Blood is thicker than water.

24- Two heads are better than one.

25- Cross the bridge when you come to it.

26- Beauty is only skin deep.

27- You can lead a horse to water but you cannot make him drink.

28- Don't count your chickens before they are hatched.

29- Don't put all your eggs in one basket.

30- Don't cry over spilt milk.

31- Every dog has its day.

32- Fortune favours the brave.

33- Great minds think alike.

34- Half a loaf is better than no bread.

35- He who laughs last laughs best.

36- Honesty is the best policy.

37- If the shoe fits, wear it.

38- It is no use crying over spilt milk.

39- Kill two birds with one stone.

40- Laugh and the world laughs with you, weep and you weep alone.

41- Let bygones be bygones.

42- Let sleeping dogs lie.

43- Make hay while the sun shines.

44- Money makes the mare go.

45- Necessity knows no law.

46- Nothing succeeds like success.

47- Out of sight, out of mind.

48- Practice what you preach.

49- Rome was not built in a day.

50- Silence is golden.

51- Strike while the iron is hot.

52- Take care of the pence and the pounds will take care of themselves.

53- The early bird catches the worm.

54- The pen is mightier than the sword.

55- There is no smoke without fire.

56- Time flies.

57- Time heals all wounds.

58- Time is money.

59- To err is human, to forgive divine.

60- Two wrongs don't make a right.

61- Where there's a will there's a way.

62- You can't have your cake and eat it too.

Assignment

Assignment No. 4

Q.1 Explain the concept of Data Flow Diagram (DFD) with suitable example.

Ans. Data Flow Diagram (DFD) is a graphical representation of the flow of data through an information system. It shows how data enters the system, how it is processed, stored, and how it leaves the system. DFD is also known as bubble chart or process model.

DFD is used to represent the functional requirements of a system. It is a tool for structured analysis and design. It helps in understanding the system at different levels of detail.

Components of DFD:

1. External Entity: It is a source or destination of data outside the system. It is represented by a rectangle.
2. Process: It is a function that transforms input data into output data. It is represented by a circle or rounded rectangle.
3. Data Store: It is a place where data is stored for later use. It is represented by two parallel lines.
4. Data Flow: It is the movement of data between entities, processes, and data stores. It is represented by an arrow.

Example: Consider a simple library system. The external entity is the Member. The processes are Issue Book and Return Book. The data store is Book Record. The data flows are Book Request, Book Issued, Book Returned, and Fine Amount.

Q.2 Explain the concept of Entity Relationship Diagram (ERD) with suitable example.

Ans. Entity Relationship Diagram (ERD) is a graphical representation of the entities and their relationships in a database. It is used to model the data and its relationships before implementing the database.

ERD is also known as ER diagram. It is a high-level conceptual model that helps in designing the database.

Components of ERD:

1. Entity: It is a real-world object or concept about which data is stored. It is represented by a rectangle.
2. Attribute: It is a property or characteristic of an entity. It is represented by an ellipse.
3. Relationship: It is an association between two or more entities. It is represented by a diamond.
4. Cardinality: It specifies the number of instances of one entity that can be related to the number of instances of another entity. It is represented by 1, M, or N.

Example: Consider a simple student database. The entities are Student and Course. The attributes of Student are Roll No, Name, and Address. The attributes of Course are Course ID and Course Name. The relationship is Enrolls. The cardinality is Many-to-Many, as one student can enroll in many courses and one course can have many students.