Module -1

(1) What is SDLC?

The Software Development Life Cycle SDLC refers to a

methodology with clearly defined processes for creating

high-quality software. in detail, the SDLC methodology

focuses on the following phases of software

development:

SDLC is a structure imposed on the development of a

software product that defines the process for planning, implementation, testing, documentation, deployment, and ongoing maintenance and support. There are a number of different development models.

 Requirement Phase

 Planning/Analysis

 Designing phase

 Software development

 Testing Phase

 Deployment

(1) Requirement analysis

Bussiness analyst collects the requirement from the

customer/client as per the clients bussiness needs and

documents the requirements in the bussiness

requirement specification and provides the same to

development Team.

(2) Analysis phase / Planning Phase

Once the requirement gathering is done the next step is

to define and document the product requirement and

get them approved from the customer.This is done

through software requirement specification document

(SRS).

Key people involved in this phase are project manager,

bussiness analyst and senior members in the Team.

The outcome of this phase is software requirement

specification.

(3) Design Phase

High level design (HLD)- It give the architecture of the

software product to be developed and is done by

architects and senior developers.

Low Level Design (LLD)- It is done by senior developers.It

describe how each and every feature in the product

should work and how every component should work.

(4) Software development

This is the phase where we start building the software

and start writing the code for the product.The outcome

from this phase is Source code Document and the

developed product.

(5) Testing phase

When the software is ready ,it is sent to the test

Team.Once the QA makes sure that the software is error

free, it goes to the next stage,which is implementation.

1. Deployment

Maintanance phaseDeployment is done by the Deployment or Implementation engineers.Once the customers start using the developed system then the actual problems will come up and needs to be solved from time to time.Maintenance should be done as per service level Agreement.

(2) What is agile methodology?

In software development, term agile means the ability

to respond to changes- changes from

Requirements,Technology and People.

It is an iterative and incremental process.

Direct collaboration with customers.

Each iteration lasts from one to three weeks.

Delivers multiple Software Increments.

Manifesto (a public declaration of policy and aims)

Agile manifesto principles

 **Individuals and interactions** - in agile development,

self-organization and motivation are

important, as are interactions like co-location and pair

programming.

 **Working software** - Demo working software is

considered the best means of communication with the

customer to understand their requirement, instead of just

depending on documentation.

 **Customer collaboration** - As the requirements cannot

be gathered completely in the beginning of the project

due to various factors, continuous customer interaction is

very important to get proper product requirements.. **Responding to change** - agile development is focused

on quick responses to change and continuous

development.

Pros

 Is a very realistic approach to software development

 Promotes teamwork and cross training.

 Functionality can be developed rapidly and

demonstrated.

 Resource requirements are minimum

 Good model for environments that change steadily.

 Minimal rules, documentation easily employed.

 Enables concurrent development and delivery within an

overall planned context.

 Little or no planning required

 Easy to manage

 Gives flexibility to developers

 Suitable for fixed or changing requirements

 Delivers early partial working solutions.

Cons

 Not suitable for handling complex dependencies.

 More risk of sustainability, maintainability and

extensibility.

 An overall plan, an agile leader and agile PM practice

is a must without which it will not work.

 Strict delivery management dictates the scope,

functionality to be delivered, and adjustments

to meet the deadlines.

 Depends heavily on customer interaction, so if

customer is not clear, team can be driven in

the wrong direction. There is very high individual dependency, since there is minimum documentation generated.

(3)What is SRS ?

 A software requirements specification (SRS) is a

complete description of the behaviour of the

system to be developed.

 It includes a set of use cases that describe all of the

interactions that the users will have with

the software.

 Use cases are also known as functional requirements.

In addition to use cases, the SRS also

contains non functional (or supplementary) requirements.

 Non-functional requirements are requirements which

impose constraints on the design or

implementation (such as performance requirements,

quality standards, or design

constraints).

**Types of Requirements**

 Requirements are categorized in several ways. The

following are common categorizations of

requirements that relate to technical management:

 Customer Requirements

 Functional Requirements

 Non-Functional Requirements

**Customer Requirements**

 The customers are those that perform the eight primary

functions of systems engineering,with special emphasis on the operator as the key

customer. Operational requirements will

define the basic need and, at a minimum, answer the

questions posed in the following

listing:

 Operational distribution or deployment: Where will the

system be used?

 Mission profile or scenario: How will the system

accomplish its mission objective?

 Performance and related parameters: What are the

critical system parameters to

accomplish the mission?

 Utilization environments: How are the various system

components to be used?

 Effectiveness requirements: How effective or efficient

must the system be in performing its mission?

 Operational life cycle: How long will the system be in

use by the user?

**Functional Requirements**

 Functional Requirements are very important system

requirements in the system design

process. These requirements are the technical

specifications, system design parameters and

guidelines,

data manipulation, data processing, and calculation

modules etc , of the proposed system.

 **For Example:** The following are the requirements of

Google Email Service

 The system shall support the ability to receive emails

 The system shall support the ability to send emails

 The system shall support the ability to create new

folders The system shall support the ability to filter emails in different folders

 The system shall support the ability to attach different

kind of attachments

 The system shall support the ability to create and

maintain address book

 The system shall support the ability to create unlimited

user accounts with different

email addresses

**Non-Functional Requirements**

 Non-functional requirements are requirements that

specify criteria that can be used to judge

the operation of a system, rather than specific behaviours.

Non-functional requirements are

qualities or standards that the system under development must have or comply with, but which are

not tasks that will be automated by the system.

 Example non-functional requirements for a system

include:

 system must run on Windows Server 2003

 system must be secured against Trojan attacks

 A software development methodology helps to identify,

document, and realize the requirements. Non functional

requirements can be divided into following categories:

 Usability

 Reliability

 Performance

 Security

1. What is oops?

 Identifying objects and assigning responsibilities to

these objects.

 Objects communicate to other objects by sending

messages.

 Messages are received by the methods of an object

 An object is like a black box.

The internal details are hidden.

 Object is derived from abstract data type

 Object-oriented programming has a web of interacting

objects, each house-keeping its own

state.

 Objects of a program interact by sending messages to

each other.

(5)Basic concept OOps?

The basic concepts of OOps are following under:

 Object

 Class

 Encapsulation

 Inheritance

 Polymorphism

 Overriding

 Overloading

 Abstraction

(6) What is objects?

 An object represents an individual, identifiable item,

unit, or entity, either real or abstract, with a well-defined

role in the problem domain. An "object" is anything to which a concept applies.

That is both data and function that operate on data

are bundled as a unit called as object.

> An object has the responsibility to *know* and the

responsibility

to *do.*

*(7)What is class?*

A class represents an abstraction of the object and

abstracts the properties and behaviour of that object **.**

.

When you define a class, you define a blueprint for an

object. .

The class is one of the Basic concepts of OOPs which is a

group of similar entities.

(7) What is Encapsulation?

Encapsulation is the practice of including in an object

everything it needs hidden from other objects. The

internal state is usually not accessible by other objects.

Encapsulation is placing the data and the functions

that work on that data in the same place. While working

with procedural languages, it is not always clear which

functions work on which variables but object-oriented

programming provides you framework to place the data

and the relevant functions together in the same object.**Encapsulation** in Java is the process of wrapping up of data (properties) and behavior (methods) of an object into a single unit; and the unit here is a Class (or

interface).

Encapsulate in plain English means *to enclose or be*

*enclosed in or as if in a capsule*. In Java, a class is the

capsule (or unit).

In Java, everything is enclosed within a class or interface,

unlike languages such as C and C++, where we can have

global variables outside classes.

Encapsulation enables data hiding**,** hiding irrelevant

information from the users of a class and exposing only

the relevant details required by the user.

1. What is Inheritance?

 Inheritance means that one class inherits the characteristics of another class. This is also called a “is a” relationship .

 One of the most useful aspects of object-oriented programming is code re usability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.

 This is a very important concept of object-oriented programming since this feature helps to

reduce the code size.

 Inheritance describes the relationship between two classes. A class can get some of its characteristics from a parent class and then add unique features of its own.

 In general, Java supports single-parent, multiple-children inheritance and multilevel inheritance (Grandparent-> Parent -> Child) for classes and interfaces. Java supports multiple inheritances (multiple parents, single child) only through interfaces.

 In a class context, inheritance is referred to as implementation inheritance, and in an interface context, it is also referred to as interface inheritance.



For example consider a Vehicle parent class and its child class Car.

 Vehicle class will have all common properties and functionalities for all vehicles in common and Car will inherit those common properties from the Vehicle class and then add those properties which are specific to a car.



Here, Vehicle is known as base class, parent class, or super class.

 Car is known as derived class, Child class or subclass.

1. What is Polymorphism?



Polymorphism means “having many forms”.

It allows different objects to respond to the same message in different ways, the response specific to the type of the object.



The most important aspect of an object is its *behaviour*(the things it can do). A behaviour is initiated by sending a *message*to the object (usually by calling a method).



The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism.

 The ability to change form is known as polymorphism.



There is two types of polymorphism in Java



Compile time polymorphism(Overloading)

 Runtime polymorphism(Overriding)

(11)What is RDBMS ?



RDBMS stands for Relational Database Management System. RDBMS is the basis for SQL, and for all modern database systems like MS SQL Server, IBM DB2, Oracle, My SQL, and Microsoft Access.

 A Relational database management system (RDBMS) is a database management system (DBMS) that is based on the relational model as introduced by E. F. Codd.



Most of today's databases are relational:

 database contains 1 or more *tables*

 table contains 1 or more *records*

 record contains 1 or more *fields*

 fields contain the data .

