**Module : 1**

**Q-1. What is SDLC?**

**Ans : SDLC** stands for **Software Development Life Cycle**. It is a structured process used for planning, creating, testing, and deploying software systems.

**Q-2. Write SDLC phases with basic introduction.**

**Ans :** SDLC stands for **Software Development Life Cycle.** Following are the phases of SDLC with basic introducation.

 **Requirement Gathering & Analysis :**

-Understand what the users need.

-Identify system requirements.

 **Planning :**Define the scope, resources, timeline, and risks.

 **Design :**Create architecture and design specifications (UI, database, system interfaces).

 **Development (Coding) :**Actual code is written based on the design.

 **Testing :**Verify the system works as expected (unit, integration, system, acceptance testing).

 **Deployment :**Release the software to users (production environment).

 **Maintenance :** Fix bugs, update features, and improve performance after release.

**Q-3. What is RDBMS?**

**Ans : RDBMS** stands for **Relational Database Management System**.

It is a type of database management system that stores data in a **structured format using rows and columns( tables or tabular format)**. The relationships between data are also stored in tables, which makes data easy to access and manage.

1 - Tabular Format - row and columns

2 - Constrain – PK (Primary Key), FK( Foreign key)

3 - Relationship - many to many , many to one, one to many

Language used in RDBMS : mysql, oracle, postgres , sqlite - sql

**Q-4. What is SQL ?**

**Ans :** SQL stands for **Structured Query Language**. The standard language used to interact with an RDBMS (e.g., SELECT, INSERT, UPDATE, DELETE).

**Q-5. Write SQL Commands ?  
Ans :**

**1. Creating a Table :**

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

hire\_date DATE,

salary DECIMAL(10, 2)

);

**2. Inserting Data :**

INSERT INTO employees (employee\_id, first\_name, last\_name, hire\_date, salary)

VALUES (1, 'John', 'Doe', '2023-01-15', 60000.00);

**3. Selecting Data :**

SELECT \* FROM employees;

**4. Updating Data :**

UPDATE employees

SET salary = 65000.00

WHERE employee\_id = 1;

**5. Deleting Data :**

DELETE FROM employees

WHERE employee\_id = 1;

**6. Join Tables :**

SELECT e.first\_name, e.last\_name, d.department\_name

FROM employees e

JOIN departments d ON e.department\_id = d.department\_id;

**Q-6. What is Intergration testing?**

**Ans : Integration testing** is a level of software testing where individual components or modules of a system are combined and tested **as a group** to verify that they work together correctly.

**Key Points:**

* It **comes after unit testing** (where individual modules are tested in isolation) and **before system testing**.
* The goal is to detect **interface defects** or issues in the interaction between integrated units.
* It checks **data flow**, **control flow**, and **communication** between modules.
* It can involve testing internal module interactions (like between classes in a library) or external ones (like between services or APIs).

**Q-7. What is Alpha testing?**

**Ans :** **Alpha testing** is a type of **internal acceptance testing** performed by the **development team or quality assurance (QA) team** **before** the product is released to real users or customers.

**Key Characteristics of Alpha Testing:**

* **Conducted in-house** — usually by developers, testers, or internal staff.
* **Still under development** — the product may have some incomplete features or known bugs.
* **Simulates real user behavior** — but in a controlled environment.
* **Advantages of Alpha testing:** Identify bugs, usability issues, and performance problems **early**, before external release.

**When is Alpha Testing Done?**

* After unit testing and integration testing are complete.
* Before **beta testing**, which involves real users outside the development team.

**Example :**

A software company develops a new messaging app. Before releasing it to a selected group of beta testers:

* The **QA team installs the app** on various devices.
* They **test sending messages**, **group chats**, **push notifications**, and simulate **user behavior**.
* They log bugs like messages not sending or the app crashing on certain devices.

**Q-8. What is Beta testing?**

**Ans :** **Beta testing** is a type of **external user acceptance testing** where a nearly finished product is released to a **limited group of real users** outside the development team to identify bugs, usability issues, and gather feedback **before the final release**.

**Key Characteristics of Beta Testing:**

* Performed by **real users in real environments** (not by developers or internal testers).
* The product is **feature-complete**, though minor bugs may still exist.
* Users provide **feedback** on performance, usability, and issues they encounter.
* **Final step** before the product goes live.

**Example :**

A company develops a new mobile app for food delivery. After internal alpha testing:

* They release the app to **a group of 1,000 beta testers**.
* These users try out the app, place real orders, and report bugs or confusing UI elements.
* Based on feedback, the company fixes bugs and improves user experience before launching publicly.

**Advantages of Beta Testing:**

* Detect bugs or issues not caught during internal testing.
* Assess product performance in **diverse user environments**.
* Gather **honest user feedback** on features, design, and usability.
* Build user trust and hype before launch.

**Q-9. What is GUI testing?**

**Ans :** **GUI Testing (Graphical User Interface Testing)** is a type of software testing that focuses on **ensuring that the user interface (UI) of an application works as expected**.

**GUI testing checks:**

* Layout and design elements (buttons, menus, icons)
* User interactions (clicks, keyboard input, gestures)
* Functional flow (navigation, forms, popups)
* Visual consistency (fonts, colors, alignments)

**Main Goals:**

* Verify that all GUI elements are present and work correctly
* Ensure user interactions produce the expected results
* Catch visual or functional bugs in the interface

**Example :**

In a **login screen**, GUI testing would check:

* Are the **username and password fields** visible and working?
* Does the **login button** trigger the correct action?
* Does the **forgot password** link work and lead to the right screen?
* Are any **error messages** displayed correctly on invalid input?

**Q-10. What is Load testing?**

**Ans :** **Load testing** is a type of **performance testing** that evaluates how a system behaves when subjected to a **specific amount of expected user traffic or data load**.

**Advantage of Load Testing:**

* To determine if a system can handle **normal and peak usage**.
* To measure **response time, stability, and resource usage** under load.
* To identify **performance bottlenecks** before the system goes live.

**Example:**

Imagine you're launching an e-commerce website for a major sale event:

* You expect **5,000 users** to be online at the same time.
* Load testing will simulate those users **browsing products, adding items to the cart, and checking out**.
* You measure if the site remains responsive or **slows down/crashes** under the load.

**Q-11. What is stress testing?**

**Ans :** **Stress testing** is a type of **performance testing** that evaluates how a system behaves **under extreme or breaking-point conditions**—**beyond normal operating capacity**.

**Advantage of Stress Testing:**

* To determine how the system handles **heavy loads, resource exhaustion**, or unexpected spikes.
* To identify the **breaking point** and ensure the system **fails gracefully** (without crashing unpredictably).
* To check if the system can **recover** after failure.

**Example :**

You're testing a stock trading platform:

* Normally it supports 10,000 users.
* In a stress test, you simulate **50,000+ users** submitting trades simultaneously.
* You observe how the system reacts: **Does it slow down, crash, or recover properly?**

**Q-12. What is White box testing and list the types of white box testing?**

**Ans :** **White box testing** (also called **clear box**, **glass box**, or **structural testing**) is a software testing technique where the **internal structure, logic, and code** of the application are **known and tested directly** by the tester.

**Advantage of White Box Testing:**

* To verify the **flow of inputs through code**.
* To ensure **all paths, conditions, and loops** behave as expected.
* To test for **logic errors**, **boundary issues**, and **code security**.

This type of testing is usually done by **developers** or testers who understand the source code.

**Key Features:**

* Requires **access to the source code**.
* Focuses on **internal logic**, not just functionality.
* Complements black box testing (which focuses only on inputs and outputs).

**Types of White Box Testing:**

**1. Unit Testing:**

* + Testing **individual functions/methods** in isolation.
  + Often automated using frameworks like **JUnit (Java)**, **PyTest (Python)**, etc.

**2. Loop Testing**

* + Checks for errors in **loops** (e.g., infinite loops, off-by-one errors).
  + Types include: simple loops, nested loops, concatenated loops.

**3. Branch Testing**

* + Ensures that **all possible branches** (e.g., if/else conditions) are executed at least once.

**4. Path Testing**

* + Tests **all possible execution paths** through the code.
  + Helps find **untested or unreachable code**.

**5. Statement Testing**

* + Ensures **every line of code (statement)** is executed at least once.
  + Measures **statement coverage**.

**6. Condition Testing**

* + Tests all **Boolean expressions** in the code to verify all logical outcomes (true/false).

**7. Security Testing (White Box Focused)**

* + Inspects the code for **vulnerabilities**, such as SQL injection points, buffer overflows, etc

**Q-13. What is Black box testing? What are the different Black box testing techniques?**

**Ans :** **Black Box Testing** is a software testing method where the **internal structure or code** of the application is **not known** to the tester. Instead, testing is based entirely on the **input-output behavior** of the system.

**Advantage of Black Box Testing:**

* To verify **functional correctness** without knowing how the software works internally.
* To test the **entire system or specific features** from a **user's perspective**.
* To find **unexpected behavior**, **incorrect outputs**, and **UI/UX issues**.

Anyone—**testers, end users, or QA engineers**—can perform black box testing, regardless of coding knowledge.

**Example :**

Imagine testing a login form:

* You enter a username and password (input).
* You expect access if the credentials are correct, or an error if not (output).
* You don’t need to know how the backend checks credentials—only whether it works correctly.

**Types of Black Box Testing Techniques:**

**1. Equivalence Partitioning**

* + Divide input data into **valid and invalid partitions**.
  + Test **one input from each partition**, assuming all will behave similarly.
  + **Example:** For age input (valid range: 18–60), test values like 17 (invalid), 25 (valid), 61 (invalid).

**2.** **Boundary Value Analysis (BVA)**

* + Focus on values at the **edges** of input ranges.
  + **Example:** If valid age is 18–60, test 17, 18, 60, 61.

**3. Decision Table Testing**

* + Create a table of **conditions and actions** to test combinations of inputs.
  + Useful for **complex business logic**.
  + Helps ensure all decision paths are covered.

**4**. **State Transition Testing**

* + Tests how the system **transitions between states** based on inputs.
  + **Example:** ATM states—Card Inserted → PIN Entered → Transaction → Eject Card.

**5. Random Testing / Monkey Testing**

* + Inputs are generated **randomly** without predefined rules.
  + Good for testing **unexpected user behavior** or **stress situations**.

**6. Error Guessing**

* + Relies on tester’s **experience** to guess likely places of failure.
  + **Example:** Testing with special characters in a name field to see if it breaks.

**Q-14. When should “ Regression Testing” be performed?**

**Ans :** **Defination :** **Regression Testing** is a type of software testing that ensures that **new code changes** do **not negatively affect** the **existing functionality** of the application.

**Advantage of Regression Testing:**

* To confirm that previously working features still work as expected after:
  + Bug fixes
  + New features
  + Code improvements or refactoring
* To catch **unexpected side effects** caused by recent changes.

**Regression Testing** should be performed **any time changes are made to the codebase**, to ensure that **existing functionality still works correctly** and **new bugs haven’t been introduced**.

**Q-15. What is 7 key principles? Explain in detail?**

**Ans :** The **7 Key Principles of Software Testing** are fundamental guidelines that help testers and teams ensure effective, efficient, and purposeful testing.

**1. Testing Shows the Presence of Defects**

**Testing can show that defects are present, but cannot prove that there are no defects.**

* The purpose of testing is to uncover bugs—not to prove the software is flawless.
* Even if no defects are found, it **doesn't guarantee** the software is 100% error-free.
* **Example:** A login system might pass all current test cases but could still fail under a new, unexpected condition.

**2. Exhaustive Testing is Impossible**

**It is not possible to test all inputs and combinations.**

* There are **too many possible input combinations, paths, and conditions** to test everything.
* Instead, testing should be **risk-based and prioritized** to focus on the most critical areas.
* **Example:** Testing every possible date input is impractical; test only typical, boundary, and invalid cases.

**3. Early Testing Saves Time and Money**

**The earlier you start testing, the more cost-effective it is.**

* Detecting and fixing bugs early (e.g., during the design or requirements phase) is **cheaper** than fixing them after release.
* Involving testers early helps clarify **requirements** and avoid miscommunication.
* **Example:** Fixing a design flaw in the planning phase costs much less than fixing it in production.

**4. Defect Clustering**

**A small number of modules contain most of the defects.**

* **80/20 Rule (Pareto Principle)**: 80% of problems come from 20% of the code.
* Focus more testing effort on **high-risk or frequently changed areas**.
* **Example:** A payment processing module in an e-commerce app might have more bugs and deserve more attention than a static "Contact Us" page.

**5. Pesticide Paradox**

**Repeating the same tests will eventually stop finding new bugs.**

* Over time, test cases become **less effective** at discovering new issues.
* To overcome this, you must **review and update** test cases regularly.
* **Example:** If you only test login with the same three usernames every time, you may miss bugs that show up with different formats or data types.

**6. Testing is Context Dependent**

**Testing should be tailored to the type and purpose of the application.**

* Different systems require different approaches:
  + A **medical device** needs thorough compliance and safety testing.
  + A **video game** may prioritize performance and user experience.
* There is **no one-size-fits-all** testing strategy.

**7. Absence-of-Errors Fallacy**

**Just because the software has no known bugs doesn’t mean it's useful.**

* Software that **doesn't meet user needs** is still a failure—even if it has no technical bugs.
* Testing should focus not only on correctness but also on **fit for purpose**.
* **Example:** A perfectly coded app that doesn't allow users to export data (a key business requirement) is still a failure.

**Q-16. What is QA v/s QC v/s tester?**

**Ans : . QA (Quality Assurance)**

**Definition**:  
**Quality Assurance** is a **process-oriented** approach focused on **preventing defects** by improving the development process.

**Goal**:  
Ensure the software development process is followed correctly to maintain quality.

**Key Activities**:

* Process definition and improvement
* Standards and guidelines enforcement
* Audits and process reviews
* Root cause analysis of defects
* Ensuring compliance with regulations (e.g., ISO, CMMI)

**Think of QA as**:  
“**Building the right process** to build the product right.”

**2. QC (Quality Control)**

**Definition**:  
**Quality Control** is a **product-oriented** approach that focuses on **identifying defects** in the final software product.

**Goal**:  
Ensure the software **meets the required quality standards** and behaves correctly.

**Key Activities**:

* Functional and non-functional testing
* Reviewing test results
* Reporting and tracking defects
* Ensuring the product meets specifications

**Think of QC as**:  
“**Checking the product** to ensure it was built right.”

**3. Tester**

**Definition**:  
A **tester** is a person who performs **testing activities** as part of **Quality Control**.

**Goal**:  
Find bugs, verify functionality, and ensure that the software behaves as expected.

**Key Activities**:

* Writing test cases and scenarios
* Executing manual or automated tests
* Reporting defects and retesting fixes
* Working closely with developers and QA teams
* Validating business requirements

**Think of a tester as**:  
“**The person who executes QC** tasks to ensure product quality.”

**Q-17. Difference between smoke and sanity?**

**Ans : Definition:**

Smoke testing is a **basic, broad-level test** performed on a software build to ensure that the **critical functionalities work** and the build is stable enough for further testing.

**Advantages:**

To **verify the stability** of a new build — “Is the app up and running?”

**When Performed:**

* **After a new build is deployed**
* Before starting **deeper functional or regression testing**

**Scope:**

* **Wide but shallow** — covers major functionalities without going into detail

**Speed:**

* **Fast** and often **automated**

**Example:**

Open the application, log in, navigate through the main screens — if it works, deeper testing can begin.

**Sanity Testing :**

**Definition:**

Sanity testing is a **focused test** to ensure a **specific bug fix or feature** is working correctly **without breaking related functionality**.

**Advantages:**

To **validate correctness** of recent changes — “Did this fix or feature work properly?”

**When Performed:**

* **After minor changes**, bug fixes, or patches
* When there is **not enough time** for full regression testing

**Scope:**

* **Narrow but deep** — focuses only on the area of change

**Speed:**

* Also fast, but more **targeted and selective**

**Example:**

After a bug fix for the "Add to Cart" button, test that feature and related cart functions only.

**Q-18. Explain types of performance testing?**

**Ans :** Performance testing is a **non-functional testing** technique used to evaluate how well an application performs under various conditions. It focuses on **speed, responsiveness, stability, scalability**, and **resource usage**.

Here are the **main types of performance testing**, each with its specific purpose

1. **Load Testing**

**Purpose**:  
To check how the system performs under **expected user load**.

**What it measures**:

* Response time
* Throughput
* Resource usage under normal and peak conditions

**Example**:  
Testing an e-commerce website with 1,000 concurrent users browsing and making purchases.

### 2. ****Stress Testing****

**Purpose**:  
To evaluate how the system performs under **extreme or beyond-normal loads**.

**What it measures**:

* Stability and error handling under overload
* System’s breaking point
* Recovery capability

**Example**:  
Simulating 10x the expected users to see if the system crashes or slows down dramatically.

### 6. ****Volume Testing (Flood Testing)****

**Purpose**:  
To test system performance when handling **large volumes of data**.

**What it measures**:

* Database performance
* Data processing capability

**Example**:  
Uploading 1 million records into the system and observing how search and retrieval perform.

**Q-19. What is join?**

**Ans :** A **JOIN** is an SQL operation used to **combine rows from two or more tables** based on a **related column** between them.

**Uses :**

* To **retrieve data** that is distributed across multiple tables.
* Useful in **relational databases** where normalization separates data into related tables.

**Syntax:**

SELECT columns

FROM table1

JOIN table2

ON table1.common\_column = table2.common\_column;

**Q-20. Write types of joins.**

**Ans :**

### 1. INNER JOIN

**Returns:** Only the rows where there is a match in **both** tables.

SELECT \*

FROM table1

INNER JOIN table2

ON table1.id = table2.id;

**Use when:** You only want records that exist in both tables.

**2. LEFT JOIN (LEFT OUTER JOIN)**

**Returns:** All rows from the **left** table, and matched rows from the right. If no match, NULLs are returned for the right table.

sql

CopyEdit

SELECT \*

FROM table1

LEFT JOIN table2

ON table1.id = table2.id;

**Use when:** You want all data from the left table, regardless of whether it has a match in the right table.

**3. RIGHT JOIN (RIGHT OUTER JOIN)**

**Returns:** All rows from the **right** table, and matched rows from the left. If no match, NULLs are returned for the left table.

sql

CopyEdit

SELECT \*

FROM table1

RIGHT JOIN table2

ON table1.id = table2.id;

**Use when:** You want all data from the right table, even if there's no match in the left.

**4. FULL JOIN (FULL OUTER JOIN)**

**Returns:** All rows from **both** tables. If there’s no match, NULLs fill the gaps.

SELECT \*

FROM table1

FULL OUTER JOIN table2

ON table1.id = table2.id;

**Use when:** You want everything from both tables, matched or not.

**Q-21. Difference between RDBMS and DBMS?**

**Ans : DBMS (Database Management System):**

* **General Purpose:** A DBMS is a software system designed to manage and organize data within a database.
* **Data Storage:** Data can be stored in various formats, including hierarchical, network, or object-oriented models, not just tables.
* **Relationships:** May or may not support relationships between data elements.
* **Query Language:** May use different query languages or methods for data manipulation.
* **Examples:** Windows Registry, XML databases.

RDBMS (Relational Database Management System):

* **Specific Model:** Uses the relational model, where data is organized into tables (relations) with rows (records) and columns (attributes).
* **Structured Data:** Data is stored in a structured, tabular format, making it easier to manage and query.
* **Relationships:** Emphasizes relationships between tables through the use of primary and foreign keys, ensuring data integrity.
* **Query Language:** Primarily uses SQL (Structured Query Language) for data manipulation.
* **Examples:** MySQL, PostgreSQL, SQL Server, Oracle.

**Q-22. Write a query to create the table in structured Query language.**

**Ans : Syntax :**

### CREATE TABLE table\_name (

### column1 data\_type [constraint],

### column2 data\_type [constraint],

### ...

### columnN data\_type [constraint]

### );

### Example ****: Create an**** employees ****table****

CREATE TABLE employees (

employee\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100) UNIQUE,

phone\_number VARCHAR(15),

hire\_date DATE,

job\_title VARCHAR(50),

salary DECIMAL(10, 2),

department\_id INT

);

**Q-23.Write a query to insert data into table.**

**Ans : Syntax :**

INSERT INTO table\_name (column1, column2, ..., columnN)

VALUES (value1, value2, ..., valueN);

CREATE TABLE students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

date\_of\_birth DATE,

email VARCHAR(100),

);

**Q-24. Write a query to drop database and table.**

**Ans : Syntax of Drop Table :**

DROP TABLE table\_name;

**Example :**

DROP TABLE students;

**Syntax of Drop Database :**

DROP DATABASE database\_name;

**Example :**

DROP DATABASE school\_db;

**Q-25. Write query to update data into table with validations.**

**Ans : Syntax:**

UPDATE table\_name

SET column1 = value1,

column2 = value2,

...

WHERE condition;

**Example :**

CREATE TABLE students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100),

enrollment\_date DATE

);

**Q-26. Write query to delete data into table with validations.**

**Ans : Syntax :**

DELETE FROM table\_name

WHERE condition;

**Example :**

CREATE TABLE students (

student\_id INT PRIMARY KEY,

first\_name VARCHAR(50),

last\_name VARCHAR(50),

email VARCHAR(100),

enrollment\_date DATE

);

**Q-27. Write query to insert new column in existing table.**

**Ans : Syntax : ALTER TABLE table\_name**

ADD column\_name data\_type [constraint];

**Example :** ALTER TABLE students

ADD status VARCHAR(10) DEFAULT 'active' NOT NULL;

**Q-28. Create new tables named seller and product apply foreign key in product table fetch data from both table using different joins.**

**Ans : Step 1: Create seller Table**

CREATE TABLE seller (

seller\_id INT PRIMARY KEY,

seller\_name VARCHAR(100),

contact\_email VARCHAR(100)

);

**Step 2: Create product Table with a Foreign Key to seller**

CREATE TABLE product (

product\_id INT PRIMARY KEY,

product\_name VARCHAR(100),

price DECIMAL(10,2),

seller\_id INT,

FOREIGN KEY (seller\_id) REFERENCES seller(seller\_id)

);

**Q-29. What is API testing?**

**Ans :** **(Application Programming Interface)** allows software applications to **communicate** with each other — like a messenger between services.

For example, when you use a food delivery app:

* UI lets you select items.
* API sends your order to the backend server.
* API Testing ensures this communication works correctly.

**Q-30. Type of API Testing?**

**Ans :**

**1. Rest API** : **REST API** stands for **Representational State Transfer Application Programming Interface**.  
It's a **web-based** architectural style for building APIs that allows different systems to communicate over **HTTP** using a stateless, scalable approach.

**Methods of Rest API testing :**

1) Get

2) Post

3) Put

4) Delete

**2. Soap API :**

**SOAP (Simple Object Access Protocol)** API is a **protocol-based** web service standard that allows communication between applications over a **network** using **XML**.

**Q-31. What is responsive testing?**

**Ans : Responsive Testing is the process of verifying that a website or web application displays and functions correctly across different devices, screen sizes, and orientations (mobile, tablet, desktop).**

**Q-32. Which types of tools are available for responsive testing?**

**Ans :** **Browser-Based Tools (Built-in)**

These are built into web browsers and help simulate different screen sizes.

**Tools:**

**Chrome DevTools -**Emulate various devices and screen resolutions.

**Firefox Developer Tools-** Responsive Design Mode, orientation switch.

**Safari Responsive Design Mode-** Test across Apple devices easily

**Q-33. What is full form of .ipa, .apk ?**

**Ans :**

**1) .APK :** full form of .apk is **android package kit.** It is the package file format used to distribute and install apps on Android devices.

**2) .ipa :** Full Form of .ipa is IOS app store package. It is the file format for iOS applications that can be installed on iOS devices.

**Q-34. What is purpose of exit criteria**

**Ans :**

**Definition:**

"Exit criteria are predefined goals or conditions that must be achieved before testing activities can be considered complete."

**Purposes of Exit Criteria :**

- Ensures that the product meets predefined **quality standards** before release.

- Prevents release with unresolved **critical bugs** or incomplete testing.

- Acts as a **milestone** to evaluate if testing goals (coverage, pass rate) are met.

- Provides a **structured way to close** the testing phase officially.

**Q-35. What is oops?**

**Ans :** OOPS, often referred to as OOP **(Object-Oriented Programming language)**, is a programming paradigm that revolves around the concept of "objects" that contain data and methods to manipulate that data. It aims to model real-world entities in software, making code more organized, reusable, and maintainable.

It is a combination of class, object, encapsulation, inheritance, polymorphism and abstraction.

**Q-36. Write Basic concepts of OOPS?**

**Ans : Following are the basic concepts of OOPS :**

1) Encapsulation

2) Inheritance

3) Polymorphism

4) Abstraction

**Q-37.What is Object?**

**Ans :** Object is a type of variable which stores multiple types of data is called a object.

**Q-38. What is class?**

**Ans :** A class is a combination of data and methods is called class.

**Syntax :**

Accessmodifier classkeyword classname

{

}

**Example :**

Public class login

{

// datatypes, variable, methods, constrain

}// scope of class

**Q-39.What is encapsulation?**

**Ans :** Bundling data (variables) and methods (functions) that operate on the data into a single unit — the **class**. In another way we will called encapsulation as data wrapping.

- Keeps data **safe from outside interference**

**-** Helps achieve **data hiding** using **access modifiers** (private, protected, public,default)

**Q-40.What is Inheritance?**

**Ans :** "Inheritance is a mechanism where one class can **inherit** fields and methods from another class, enabling **code reuse** and **hierarchical relationships**."Or Inheritance is a mechanism where child class uses the functionality of parent class using extend keyword.

**Uses of Inheritance :**

1) **Code reusability :** Write common code once and reuse it.

2) **Code optimization :** Code optimization is the process of modifying code to improve its efficiency, either by reducing resource consumption or by enhancing performance.

**Q-41. What is Polymorphism?**

**Ans :** Polymorphism is the context of object-oriented programming, is the ability of a single interface to represent multiple underlying forms or implementations. It allows objects of different classes to be treated as objects of a common type, enabling code reusability and flexibility. Essentially, it means that a single method or function can behave differently based on the object it is interacting with. OR One interface and multiple implementation is known as Polymorphism.

**There are two ways of Polymorphism :**

**1. Compile-time Polymorphism (Method Overloading)**

In method of overloading polymorphism there is one class multiple methods and same name but data is different.

**2. Runtime Polymorphism (Method Overriding)**

In method of Overriding polymorphism there is one class multiple methods and same name and data is same.

**Q-42. What is software testing?**

**Ans :** **Software :** Software is a collection of code that performs our instructions.

**Software** **testing :**

Software testing is the process of evaluating a software product or application to determine if it meets specified requirements and functions as expected. It involves executing a software application to identify defects, bugs, and other issues before the software is released to users. Testing ensures the software is reliable, efficient, and meets user expectations.

**Q-43. What is the difference between STLC( Software Testing Life Cycle) and SDLC ( Software Development Life Cycle) ?**

**Ans :**

**SDLC :**

-SDLC stands for Software Development Life Cycle.

- A Process followed by Developing Software, covering the full lifestyle from

requirements to maintenance.

-SDLC objective is to build a functional and reliable product.

-Phases of SDLC is

-Requirement gathering

-Design

-Development

-Testing

-Deployment

-Maintenance

-SDLC performed by developer, architecture, business analysts, etc

**STLC :**

-STLC stands for Software Testing Life Cycle.

-A Process followed for testing software, ensuring quality and correctness before release.

- STLC Objective is to ensure the product is defect-free and fulfills the requirements.

-Phases of STLC is

- Requirement Analysis

-Test planning

-Test case Execution

-Environment setup

- Test case

- Test Closure

-STLC performed by tester and QA(Quality Assurance) analyst.