Java

What are pillers of oops and explain

Difference between == and equals()

Difference between string stringbuffer , stringbuilder

String are immutable and Array are mutable why?

Difference between abstract and interface?

What are constructor? Is constructor can be overloaded or overrided?

Can constructor be static?

Explain super, final, static keyword.

Access modifier in java.

Exception in java and how you will handle exception

Difference between list and set and map

Which one you will prefer arraylist and linkedlist

Comparator and Camparable interface

What are types of stream?

How you will initiatiate , run, stop the thread?

What is memory allocation in java

What is JRE, JDK, JVM

Design pattern creational➔ singleton and pagefactory

❓ What are the pillars of OOP?

The four pillars of Object-Oriented Programming (OOP) are:

1. **Encapsulation**
    - Wrapping data (variables) and methods (functions) into a single unit (class).
    - Access modifiers (private, public) help hide internal details.
    - Example: A class with private fields and public getters/setters.
2. **Abstraction**
    - Hiding implementation details and showing only essential features.

- Achieved using abstract classes and interfaces.
- Example: A car's interface lets you drive without knowing the engine mechanics.

3. **Inheritance**
    - Allows one class to inherit fields and methods from another.
    - Promotes code reuse and hierarchical classification.
    - Example: class Dog extends Animal.
4. **Polymorphism**
    - Ability to take many forms.
    - Compile-time (method overloading) and runtime (method overriding).
    - Example: A parent class reference calling overridden methods in child classes.

Sources:

❓ Difference between == and equals() in Java

- == compares **references** (memory addresses).
- equals() compares **values/content**.
- Example:

java

```
String a = new String("hello");
String b = new String("hello");
System.out.println(a == b);      // false
System.out.println(a.equals(b));  // true
```

❓ Difference between String, StringBuffer, and StringBuilder

Feature	String	StringBuffer	StringBuilder
Mutability	Immutable	Mutable	Mutable
Thread Safety	Not thread-safe	Thread-safe	Not thread-safe

Feature	String	StringBuffer	StringBuilder
Performance	Slow (due to immutability)	Slower (due to synchronization)	Faster (no sync)

Use StringBuilder for single-threaded performance, StringBuffer for thread-safe operations.

❓ Why are Strings immutable and Arrays mutable?

- **Strings** are immutable because:
  - They are cached in the String pool.
  - Immutable objects are thread-safe.
  - Security: used in class loading, file paths, etc.
- **Arrays** are mutable because:
  - They are data structures meant to store and update values.
  - You can change elements at specific indices.

❓ Difference between abstract class and interface

Feature	Abstract Class	Interface
Methods	Can have abstract and concrete methods	Only abstract methods (until Java 8)
Variables	Can have instance variables	Only constants (static final)
Inheritance	Single inheritance	Multiple inheritance
Constructor	Can have constructors	Cannot have constructors

Use abstract class when you want partial implementation; use interface for full abstraction.

❓ What are constructors? Can they be overloaded or overridden?

- **Constructor**: Special method to initialize objects.
- **Overloading**: Yes, constructors can be overloaded (same name, different parameters).
- **Overriding**: No, constructors cannot be overridden because they are not inherited.

❓ Can a constructor be static?

No. Constructors are used to create instances, while static belongs to the class. You cannot have a static constructor in Java.

❓ Explain super, final, and static keywords

- super: Refers to parent class. Used to access parent methods/constructors.
- final: Prevents modification.
    - Final variable → constant
    - Final method → cannot be overridden
    - Final class → cannot be inherited
- static: Belongs to class, not instance.
    - Static methods/variables can be accessed without creating an object.

❓ Access modifiers in Java

Modifier	Class	Package	Subclass	World
public	✓	✓	✓	✓
protected	✓	✓	✓	✗
default	✓	✓	✗	✗
private	✓	✗	✗	✗

❓ Exception in Java and how to handle it

❓ What is an Exception in Java?

An **exception** is an **unexpected event** that disrupts the normal flow of a program during execution. It occurs when something goes wrong—like dividing by zero, accessing a null object, or trying to open a file that doesn't exist.

Java handles exceptions using a robust **exception-handling mechanism** that helps maintain program stability and prevent crashes.

🧠 Why Use Exception Handling?

- To gracefully handle runtime errors.
- To separate error-handling logic from regular code.
- To maintain program flow and avoid abrupt termination.

🧱 Types of Exceptions in Java

Java exceptions are broadly categorized into **three types**:

1. Checked Exceptions

These are exceptions that are **checked at compile time**. You must handle them using try-catch or declare them using throws.

Examples:

- IOException
- SQLException
- FileNotFoundException

Use Case:

java

```
try {
   FileReader file = new FileReader("data.txt");
} catch (FileNotFoundException e) {
   System.out.println("File not found!");
}
```

2. Unchecked Exceptions

These are **not checked at compile time**. They occur due to programming errors and are subclasses of RuntimeException.

Examples:

- NullPointerException
- ArrayIndexOutOfBoundsException
- ArithmeticException

Use Case:

java

```
int[] arr = new int[3];
System.out.println(arr[5]); // ArrayIndexOutOfBoundsException
```

3. Errors

These are serious issues that a program **should not try to handle**. They are subclasses of Error.

Examples:

- OutOfMemoryError
- StackOverflowError
- VirtualMachineError

Note: Errors indicate problems with the JVM or system resources.

🔧 How to Handle Exceptions

Java provides three main constructs:

java

```
try {
   // risky code
} catch (ExceptionType e) {
   // handling code
} finally {
   // cleanup code (optional)
}
```

- try: Wraps code that might throw an exception.
- catch: Catches and handles the exception.
- finally: Executes regardless of exception (used for cleanup).

❓ Difference between List, Set, and Map

Feature	List	Set	Map
Duplicates	Allowed	Not allowed	Keys not allowed
Order	Maintained	Not guaranteed	Key-value pairs
Example	ArrayList	HashSet	HashMap

❓ Which one to prefer: ArrayList vs LinkedList

- **ArrayList**:
    - Faster for random access.

    - Backed by array.
- **LinkedList**:
    - Faster for insert/delete.
    - Doubly linked list.

Prefer ArrayList for frequent access, **LinkedList** for frequent insertions/deletions.

❓ Comparator vs Comparable interface

Features	Comparable	Comparator
Definition	It defines natural ordering within the class.	It defines external or custom sorting logic.
Method	compareTo()	compare()
Implementation	It is implemented in the class.	It is implemented in a separate class.
Sorting Criteria	Natural order sorting	Custom order sorting
Usage	It is used for a single sorting order.	It is used for multiple sorting orders.

```
class Student implements Comparable<Student> {
    int marks;
    Student(int marks) { this.marks = marks; }

    public int compareTo(Student s) {
        return this.marks - s.marks; // ascending order
```

```
    }
}
```

Usage:

```
java
Collections.sort(studentList);
```

```
class Student {
    int marks;
    Student(int marks) { this.marks = marks; }
}

class MarksComparator implements Comparator<Student> {
    public int compare(Student s1, Student s2) {
        return s2.marks - s1.marks; // descending order
    }
}
```

Usage:

```
java
Collections.sort(studentList, new MarksComparator());
```

❓ Types of Streams in Java

📝 1. Java I/O Streams

Java I/O (Input/Output) streams are used to read and write data (bytes or characters). They are divided into two main categories:

- 🔹 **Byte Streams**

- Handle binary data (images, audio, etc.)
- Classes: InputStream, OutputStream and their subclasses
- Examples:
    - FileInputStream
    - FileOutputStream
    - BufferedInputStream
    - BufferedOutputStream

🔹 **Character Streams**

- Handle character data (text files)
- Classes: Reader, Writer and their subclasses
- Examples:
    - FileReader
    - FileWriter
    - BufferedReader
    - BufferedWriter

🔹 **Buffered Streams**

- Improve performance by reducing disk access
- Wrap around byte or character streams
- Examples:
    - BufferedReader, BufferedWriter
    - BufferedInputStream, BufferedOutputStream

🔹 **Data Streams**

- Read/write primitive data types in a machine-independent way
- Classes:
    - DataInputStream
    - DataOutputStream

🔹 **Object Streams**

- Used for object serialization/deserialization

- Classes:
    - ObjectInputStream
    - ObjectOutputStream

🌊 2. Java 8 Streams API

Java 8 introduced the **Streams API** to process collections of objects in a functional style.

🔹 Types of Java 8 Streams

Stream Type	Description
Stream<T>	For object collections (List, Set, etc.)
IntStream	For primitive int values
LongStream	For primitive long values
DoubleStream	For primitive double values

🔹 Stream Operations

- **Intermediate**: map(), filter(), sorted()
- **Terminal**: collect(), forEach(), reduce()

🔹 Example:

java

```
List<String> names = Arrays.asList("Alice", "Bob", "Charlie");
names.stream()
   .filter(name -> name.startsWith("A"))
   .forEach(System.out::println);
```

✅ Natural Order (Comparable)

Java Copy

```java
List<String> names = Arrays.asList("Charlie", "Alice", "Bob");

List<String> sortedNames = names.stream()
    .sorted() // uses natural alphabetical order
    .collect(Collectors.toList());

System.out.println(sortedNames); // [Alice, Bob, Charlie]
```

✅ Custom Order (Comparator)

Java Copy

```java
List<Integer> numbers = Arrays.asList(5, 2, 9, 1);

List<Integer> sortedDesc = numbers.stream()
    .sorted((a, b) -> b - a) // descending order
    .collect(Collectors.toList());

System.out.println(sortedDesc); // [9, 5, 2, 1]
```

❓ How to initiate, run, and stop a thread?

✅ 1. Initiate a Thread

You can create a thread in two main ways:

🔹 By Extending Thread Class

java

```
class MyThread extends Thread {
   public void run() {
      System.out.println("Thread is running");
   }
}
```

🔹 By Implementing Runnable Interface

java

```
class MyRunnable implements Runnable {
  public void run() {
    System.out.println("Thread is running");
  }
}
```

✅ **2. Run a Thread**

To run the thread, you must call start()—**not** run() directly.

🔹 **Using Thread Class**

java

```
MyThread t1 = new MyThread();
t1.start(); // starts a new thread
```

🔹 **Using Runnable**

java

```
Thread t2 = new Thread(new MyRunnable());
t2.start(); // starts a new thread
```

🔸 start() creates a new thread and calls run() internally. 🔸 Calling run() directly will execute in the current thread—not a new one.

✅ **3. Stop a Thread**

Java discourages using Thread.stop() (it's deprecated and unsafe). Instead, use **interruption or flags**.

🔹 **Using a Flag**

java

```
class MyTask implements Runnable {
  private volatile boolean running = true;

  public void run() {
    while (running) {
      System.out.println("Working...");
```

```
    }
  }

  public void stop() {
    running = false;
  }
}
```

Usage:

```
java
MyTask task = new MyTask();
Thread t = new Thread(task);
t.start();

// Later...
task.stop(); // Gracefully stops the thread
```

🔹 **Using interrupt()**

```
java
Thread t = new Thread(() -> {
  while (!Thread.currentThread().isInterrupted()) {
    System.out.println("Running...");
  }
});
t.start();

// Later...
t.interrupt(); // Signals the thread to stop
```

🧠 **Summary**

Action Method Used

Initiate new Thread() or Runnable

Run start()

Stop interrupt() or custom flag

❓ What is memory allocation in Java?

- **Heap**: Stores objects.
- **Stack**: Stores method calls and local variables.
- **Method Area**: Stores class metadata.
- **Garbage Collector**: Frees unused memory.

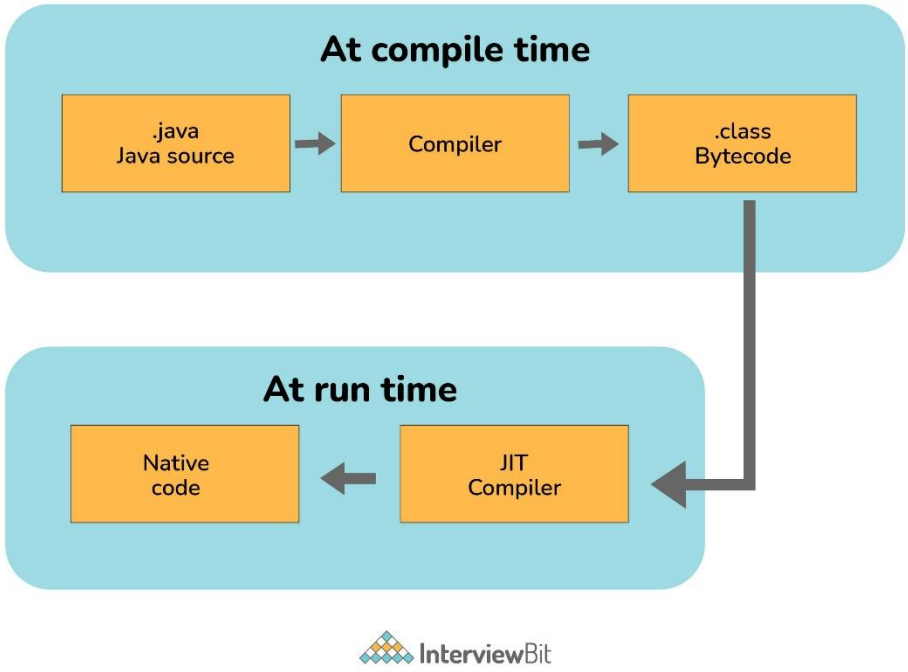
❓ What is JRE, JDK, JVM?

- **JDK**- For making java programs, we need some tools that are provided by JDK (Java Development Kit). JDK is the package that contains various tools, Compiler, Java Runtime Environment, etc.
- **JRE** - To execute the java program we need an environment. (Java Runtime Environment) JRE contains a library of Java classes + JVM. **What are JAVA Classes?** It contains some predefined methods that help Java programs to use that feature, build and execute. ***For example*** - there is a system class in java that contains the print-stream method, and with the help of this, we can print something on the console.
- **JVM** - (Java Virtual Machine) JVM is a part of JRE that executes the Java program at the end. Actually, it is part of JRE, but it is software that converts bytecode into machine-executable code to execute on hardware.

10. Tell us something about JIT compiler.

- JIT stands for Just-In-Time and it is used for improving the performance during run time. It does the task of compiling parts of byte code having similar functionality at the same time thereby reducing the amount of compilation time for the code to run.
- The compiler is nothing but a translator of source code to machine-executable code. But what is special about the JIT compiler? Let us see how it works:
    - First, the Java source code (.java) conversion to byte code (.class) occurs with the help of the javac compiler.

- Then, the .class files are loaded at run time by JVM and with the help of an interpreter, these are converted to machine understandable code.
- JIT compiler is a part of JVM. When the JIT compiler is enabled, the JVM analyzes the method calls in the .class files and compiles them to get more efficient and native code. It also ensures that the prioritized method calls are optimized.
- Once the above step is done, the JVM executes the optimized code directly instead of interpreting the code again. This increases the performance and speed of the execution.



❓ Singleton and PageFactory

- **Singleton**:
  - Ensures one instance of a class.
  - Example:

java

```
public class Singleton {
   private static Singleton instance = null;
   private Singleton() {}
   public static Singleton getInstance() {
```

```
    if (instance == null) {
      instance = new Singleton();
    }
    return instance;
  }
}
```

- **PageFactory**:
    - Used in Selenium for initializing web elements.
    - Example:

java

```
@FindBy(id="login") WebElement loginBtn;
PageFactory.initElements(driver, this);
```

Selenium

Q. What is selenium and component of selenium

1. What is Selenium? •Selenium is an open-source automation testing tool • It helps to automate web browsers • It helps to validate web applications across different browsers like Chrome, Firefox, Internet Explorer and Safari • It helps to validate the application in different platforms/Operating Systems (OS) like Windows, iOS and Linux • It supports multiple programming languages like Java, C#, Python etc.

• WebDriver is an interface • It directly communicating with browser • It helps to execute your script into multiple browsers and multiple OS • It helps to create script in multiple languages. You can choose your own language to create Selenium script

• Selenium Integrated Development Environment (IDE): It is a Firefox/Chrome plug-in that is developed to speed up the creation of automation scripts by recording the user actions on the web browser and exporting them as a reusable script.

• Selenium Remote Control (RC): It is a server that enables users to generate test scripts in their preferred programming language. It accepts commands from the test scripts and sends them to the browser as Selenium core JavaScript commands, for the browser to behave accordingly

• Selenium Grid: It allows parallel execution of tests on different browsers and operating systems by distributing commands to other machines simultaneously

Q. Difference between findelement and findelements

• findElement() → Returns the first matching WebElement, throws

NoSuchElementException if not found.

• findElements() → Returns a List of WebElements. If not found, returns empty list (size 0), no exception.

Feature	findElement	findElements
Return Type	Single WebElement	List<WebElement>
When Used	To locate **one** element	To locate **multiple** elements
If Not Found	Throws NoSuchElementException	Returns an **empty list**
Access Pattern	Directly interact with the element	Iterate over the list to interact
Example	driver.findElement(By.id("username"))	driver.findElements(By.tagName("input"))

Q. Types of wait and which wait is preferred?

1. Implicit Wait

- Applies globally to all elements.
- Tells WebDriver to wait for a certain amount of time before throwing NoSuchElementException.
- Syntax:

java

driver.manage().timeouts().implicitlyWait(Duration.ofSeconds(10));

2. Explicit Wait

- Waits for a specific condition to be met (e.g., element to be clickable).

- More flexible and targeted.
- Syntax:

java

```
WebDriverWait wait = new WebDriverWait(driver, Duration.ofSeconds(10));
wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("username")));
```

3. Fluent Wait

- Advanced form of Explicit Wait.
- Allows polling frequency and exception handling.
- Syntax:

java

```
Wait<WebDriver> wait = new FluentWait<>(driver)
  .withTimeout(Duration.ofSeconds(30))
  .pollingEvery(Duration.ofSeconds(5))
  .ignoring(NoSuchElementException.class);
```

```
WebElement foo = wait.until(new Function<WebDriver, WebElement>() {
 public WebElement apply(WebDriver driver) {
  return driver.findElement(By.id("foo"));
 }
});
```

✅ **Preferred Wait: Explicit Wait**

- It gives **precise control** over conditions.
- Ideal for **dynamic elements** and **AJAX-based applications**.
- Avoids unnecessary delays compared to Implicit Wait.

What is polling time in wait?

Polling time in Selenium refers to the **interval between successive checks** for a condition to be met during a wait operation — especially in **FluentWait**.

🔁 **What Happens During Polling?**

When you use a wait like FluentWait, Selenium:

1. Starts a timer.
2. Repeatedly checks (polls) if the condition is true.
3. Waits for the polling interval between each check.
4. Stops when the condition is met or the timeout is reached.

🧪 **Example in Java**

java

```
Wait<WebDriver> wait = new FluentWait<>(driver)
   .withTimeout(Duration.ofSeconds(30))      // total wait time
   .pollingEvery(Duration.ofSeconds(5))      // polling interval
   .ignoring(NoSuchElementException.class);


wait.until(ExpectedConditions.visibilityOfElementLocated(By.id("username")));
```

In this example:

- Selenium checks every **5 seconds** for the element to be visible.
- It will keep checking until **30 seconds** have passed or the element appears.

🧠 **Why Polling Matters**

- Reduces CPU usage compared to constant checking.
- Gives flexibility to handle dynamic content.
- Helps avoid flaky tests by waiting intelligently.

Types of dropdown in selenium how you handle it

Difference between driver.close() and driver.quit()

What is page object model and when it is used?

How you handle dyanamic web elements

Screenshot in webdriver

Annotation in TestNG

Exception in selenium

Frames

Mouse over action

How you will handle broken link

How will upload file to selenium webdriver

Limitation of selenium

What is Webdrivermanager

Select two rows of table simultaneously

What is JavascriptExecuter and when to use it

What is dataprovider

Switching between tabs

dragAndDrop

How you open website in new window

Assertion in selenium

/ & //

What is difference between a tag href tag

MySQL

What is mysql?

Difference between database and database management system

Datatype in mysql

Primary key and foreign key

What is difference between varchar and char

When where clause is used

Explain order by and group by

What is index

Types of joins

What is transaction and stored procedure

What is difference between delete , truncate and drop and when to use

How you optimize your query?

Functional testing

Types of testing

Difference between regression and retest 28

Regression Testing

Regression testing is the process of testing a software application to ensure that new changes (like updates, bug fixes, or new features) do not negatively impact the existing functionality of the software.

Example: Imagine you have a mobile app that allows users to log in, view their profile, and send messages. If a developer adds a new feature, such as the ability to upload profile pictures, regression testing ensures that after this change, users can still log in, view their profiles, and send messages without any issues

Re-Testing

Whenever the developer fixed a bug, tester will test the bug fix is called Re-testing. • Tester close the bug if it worked otherwise re-open and send to developer.

Re-Testing Vs Regression Testing

• An Application Under Test has three modules namely Admin, Purchase and Finance.

• Finance module depends on Purchase module.

• If a tester found a bug on Purchase module and posted. Once the bug is fixed, the tester needs to do Retesting to verify whether the bug related to the Purchase is fixed or not and also tester needs to do Regression Testing to test the Finance module which depends on the Purchase module

What is exploratory testing 31

Exploratory Testing

• We have to explore the application ,understand completely and test it.

• Understand the application , identify all possible scenarios , document it then use it for

testing.

• We do exploratory testing when the Application ready but there is no requirement.

• Test Engineer will do exploratory testing when there is no requirement.

Drawbacks:

• You might misunderstand any feature as a bug (or) any bug as a feature since you do not

have requirement.

• Time consuming

• If there is any bug in application , you will never know about it

Priority and severity -51

What is user acceptance testing

✅ What is UAT?

User Acceptance Testing is where end users or clients test the software to ensure it meets their business requirements and is ready for deployment.

🔍 Key Features of UAT:

Performed by: Actual users or stakeholders, not developers or testers.

Environment: Typically done in a staging environment that mimics production.

Goal: Validate that the system works as expected in real-world scenarios.

Focus: Business functionality, usability, and compliance with requirements.

🧪 Alpha Testing

Purpose: To catch bugs and usability issues before releasing the software to external users.

Key Features:

Performed by: Internal teams—developers, QA testers, or product managers.

Environment: Controlled lab-like setting, not real-world usage.

Timing: Happens before beta testing, after system testing.

Focus: Stability, core functionality, and major issues

Techniques Used: Both white-box and black-box testing.

🌍 Beta Testing

Purpose: To validate the software in real-world conditions and gather user feedback.

Key Features:

Performed by: External users—customers, industry experts, or public testers.

Environment: Real-world usage scenarios.

Timing: Happens after alpha testing, just before official release.

Focus: Minor bugs, user experience, compatibility, and performance.

Techniques Used: Mostly black-box testing.

Defect life cycle

🐞 **Simple Steps in the Defect Life Cycle**

Here's how a defect typically moves through its life:

1. **New** The defect is reported and logged for the first time.
2. **Assigned** The defect is given to a developer or team to investigate.
3. **Open** The developer starts working on fixing the defect.
4. **Fixed** The developer has made changes to fix the issue.
5. **Retest** The tester checks if the fix works correctly.
6. **Verified** The tester confirms the defect is resolved.
7. **Closed** The defect is officially marked as closed.

🔁 **Other Possible States**

- **Rejected**: If the defect is not valid or not reproducible.
- **Deferred**: If the defect is postponed to be fixed later.
- **Duplicate**: If the defect is already reported.

What is entry and exit criteria in project

🚪 Entry Criteria

Definition: Conditions that must be fulfilled before testing can begin.

Purpose: To ensure the testing team has everything needed to start testing effectively.

Examples:

Requirements and design documents are approved.

Test environment is set up and stable.

Test data is prepared.

Test cases are written and reviewed.

Code is unit tested and ready for integration/system testing.

🏁 Exit Criteria

Definition: Conditions that must be satisfied to conclude testing.

Purpose: To confirm that testing goals have been met and the product is ready for the next phase or release.

Examples:

All planned test cases are executed.

Defects are resolved or deferred with justification.

Test coverage meets the required threshold.

No critical or high-severity bugs remain.

Test summary report is completed and signed off.

SDLC

What is agile methodologies ?

What is Agile Methodology?

- It is an **Iterative** and **Incremental** Approach.
- **Iterative** means same process repeating again and again.(The process keeps on repeating).
- **Incremental** means, modules/features keep on adding on top of existing software.
- Agile is Iterative and Incremental model where requirements keeps on changing.
- As a company we should be flexible to accept requirements change, develop, test and finally release a peace of working software within short span of time.
- There will be good communication between Customer, Business Analyst, Developers & Testers.
- The Goal of the agile model is the customer satisfaction by delivering the piece of the software to the customer within short span of time.
- Agile Testing is type of testing where we follow the agile principles.

Principles of Agile

- Customer satisfaction
- Face to face communication
- Sustainable development
- Continuous feedback
- Quick respond to changes
- Successive improvement
- Self-organized
- Error-free clean node
- Collective work

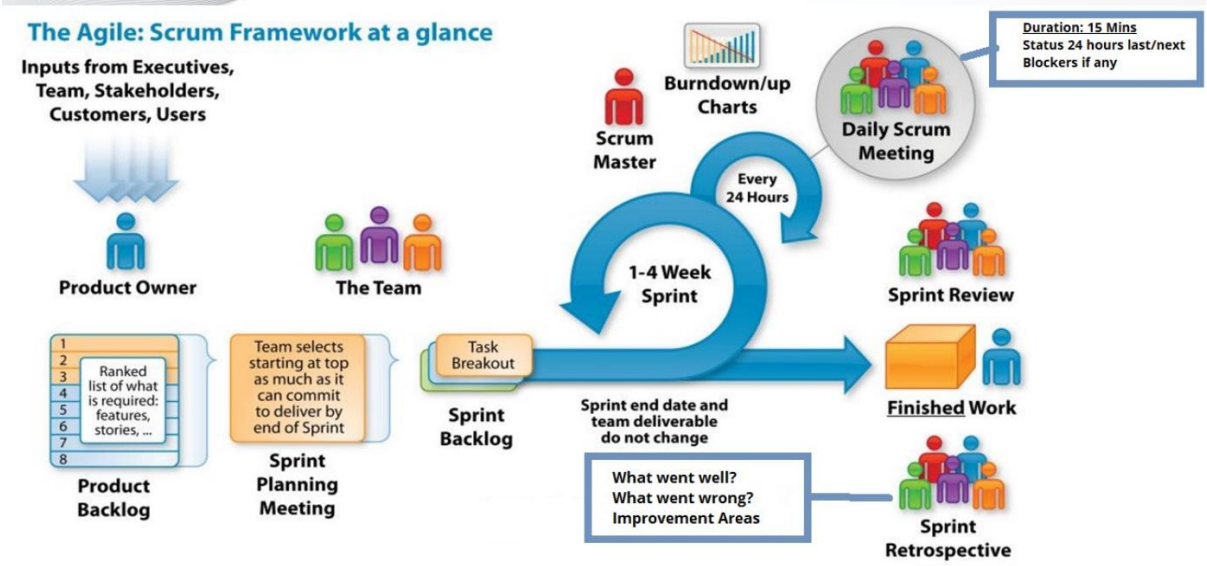
Agile frameworks

- Kanban
- Scrum
- Extreme Programming (XP)
- Crystal
- Dynamic Systems Development Method (DSDM)
- Feature-Driven Development (FDD)
- Adaptive Software Development (ASD)
- Lean Software Development (LSD)
- Scaled Agile Framework (SAFe)
- Rapid Application Development (RAD)

What is scum ?

What is Scrum?

- **Scrum** is a framework through which we build software product by following Agile Principles.
- Scrum includes group of people called as Scrum team.
  - Product Owner
  - Scrum Master
  - Dev Team
  - QA Team
- **Product Owner :**
  - Define the features of the product
  - Prioritize features according to market value
  - Adjust features and priority every iteration, as needed
  - Accept or reject work results.
- **Scrum Master:**
  - The main role is facilitating and driving the agile process.
- **Developers and QA:**
  - Develop and Test the software.



Scrum Terminology

- **User Story :** A Feature/module in a software
- **Epic :** Collection of user stories.
- **Product backlog :** Contains list of user stories. Prepared by product owner.
- **Sprint :** Period of time to complete the user stories, decided by the product owner and team, usually 2-4 weeks of time.
- **Sprint planning meeting:** Meating conducts with the team to define what can be delivered in the sprint and duration.
- **Sprint backlog :** List of committed stories by Dev/QA for specific sprint.

Scrum Terminology

- **Scrum meeting :** Meating conducted by Scrum Master everyday 15 mins. Called as Standup meeting.
  - What did you do yesterday?
  - What will you do today?
  - Are there any impediments in your way?
- **Sprint retrospective meeting :** Review meeting after completion of sprint. The entire team, including both the ScrumMaster and the product owner should participate.
- **Story point :** Rough estimation of user stories, will be given by Dev & QA in the form of Fibonacci series.
- **Burndown chart :** Shows how much work remining in the sprint. Maintained by the scrum master daily.

Agile Meetings

1) Sprint Planning:

- Attendees: Entire team (developers, testers, product owner).
- When: At the beginning of each sprint.
- Duration: Typically 1-2 hours.
- Purpose: Plan and prioritize tasks for the upcoming sprint.

2) Daily Standup (Daily Scrum):

- Attendees: Entire team.
- When: Daily, preferably in the morning.
- Duration: 15 minutes or less.
- Purpose: Share updates on work, discuss challenges, and align for the day.

3) Sprint Review:

- Attendees: Team, stakeholders, product owner.
- When: At the end of each sprint.
- Duration: 2-4 hours.
- Purpose: Showcase completed work, gather feedback, and discuss what's next.

4) Sprint Retrospective:

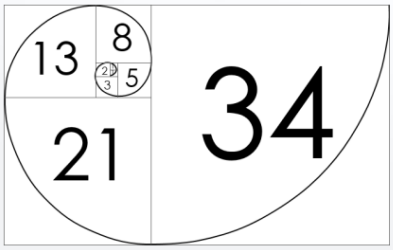
- Attendees: Team members.
- When: At the end of each sprint, after the sprint review.
- Duration: 1-2 hours.
- Purpose: Reflect on the sprint, discuss what went well and what could be improved, and plan for adjustments.

5) Backlog Grooming (Refinement):

- Attendees: Product owner, Scrum Master, development team.
- When: As needed between sprints.
- Duration: Typically 1-2 hours.
- Purpose: Review and refine the product backlog, ensuring items are well-defined and ready for upcoming sprints.

Story Point

- A **story point** is a unit of measure used to estimate the difficulty or complexity of a task or user story.
- Estimating a user story in Agile involves assigning it a story point value, and teams often use the Fibonacci sequence (1, 2, 3, 5, 8, 13, etc.)



Estimating a story using story point

1 Story Point: Typically takes a few hours to complete (half a day).

2 Story Points: Could take a day to a day and a half.

3 Story Points: Might take two days.

5 Story Points: Could take around three days.

8 Story Points: A larger task, likely taking a week.

13 Story Points: A significant effort, possibly spanning multiple weeks.

Burn-Down Charts

There are four popularly used burn down charts in Agile.

- **Product burndown chart :** A graph which shows how many Product Backlog Items (User Stories) implemented/not implemented.
- **Sprint burndown chart :** A graph which shows how many Sprints implemented/not implemented by the Scrum Team.
- **Release burndown chart :** A graph which shows List of releases still pending, which Scrum Team have planned.
- **Defect burndown chart :** A graph which shows how many defects identified and fixed.

What is sprint ?

- **Sprint :** Period of time to complete the user stories, decided by the product owner and team, usually 2-4 weeks of time.

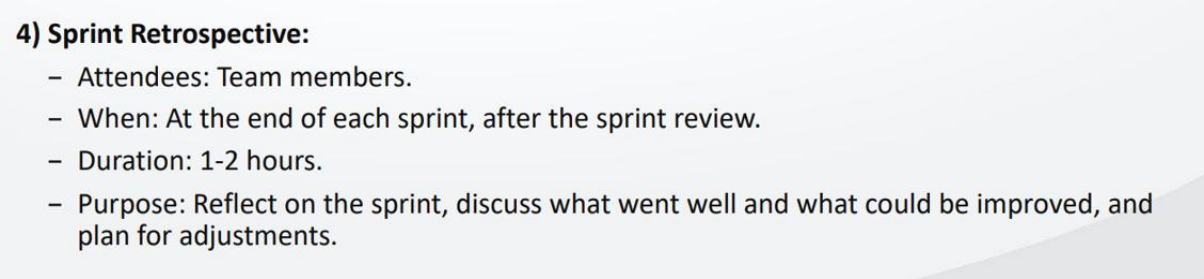
What is duration of sprint meeting ?

Sprint Planning: To decide what work will be done in the upcoming sprint and how to achieve it.

Sprint Planning	Plan work for the upcoming sprint	Up to **2 hours** per week of sprint (≈4 hours)

What is Sprint retrospection ?

Sprint Retrospective: To reflect on the sprint and find ways to improve teamwork and processes.



Sprint Retrospective Reflect and improve team processes 1–1.5 hours

What is user stories ?

They’re short, simple descriptions of a feature or functionality written from the perspective of the end user. The goal is to capture what the user wants to achieve and why, without diving into technical details.

What is test scenarios ?

Test scenarios are high-level descriptions of what to test in a software application. They represent specific situations or functionalities that need to be verified to ensure the system behaves as expected.

What is deferred testing ?

Deferred testing means **delaying a test** because you're not ready to run it yet.

📌 **Simple Example:**

Imagine you're building a mobile app that lets users pay with a credit card.

You want to test the payment feature, but:

- The payment system isn’t connected yet.
- Or the developer hasn’t finished that part.

So, you **wait** and test it **later** when everything is ready.

That’s called **deferred testing** — you’re **postponing** the test until the right time.

What is root cause analysis ?

Root Cause Analysis means digging deep to find the **main cause** of a problem so you can fix it properly and prevent it from happening again.

📌 **Simple Example:**

Let's say your mobile app keeps crashing.

You could ask:

- What happened? → The app crashed.
- Why did it crash? → Because it ran out of memory.
- Why did it run out of memory? → Because a background process didn't stop.
- Why didn't it stop? → Because the code didn't handle it correctly.

👉 The **root cause** is: the code didn't stop the background process.

Software testing lifecycle ---- 43 page

What is test environment ?

• Test Environment is a platform specially built for test case execution on the software product. • It is created by integrating the required software and hardware along with proper network configurations. • Test environment simulates production/real time environment. • Another name of test environment is Test Bed.

Boundry value analysis -37

Contents in defect reporting

Defect Report Contents

• Defect ID: Unique identification number.

• Defect Description: Detailed info, module where defect found.

• Version: Version of application where defect found.

• Steps: Detailed steps with screenshots to reproduce defect.

• Date Raised: When defect is raised.

• Reference: Provide reference to requirements, design, architecture, screenshots.

• Detected By: Tester who raised defect.

• Status: Status of defect.

• Fixed By: Developer who fixed it.

- Date Closed: When defect closed.
- Severity: Describes impact.
- Priority: Related to fixing urgency. (High/Medium/Low).