PRATIXA MISTRY

Module–1(Fundamental)

1) What is SDLC?

SDLC is a structure of 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.
A Software Development Life Cycle is series of steps, or phases, that provide a model for the development and lifecycle management of piece of software.

2) What is software testing?

Software Testing is a process used to identify the correctness, completeness, and quality of developed computer software.

Test activities also exist before and after test execution.

It can also be stated as the process of validating and verifying that a software product meets the business and technical requirements and works as expected and can be implemented with the same characteristic.

3) What is agile methodology?

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.

Agile Methods break the product into small incremental builds.

These builds are provided in iterations.

Each iteration typically lasts from about one to three weeks.

Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

At the end of the iteration a working product is displayed to the customer and important stakeholders.

4) What is SRS

A software requirements specification (SRS) is a complete description of the behavior 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 nonfunctional requirements.

Non-functional requirements are requirements which impose constraints on the design or implementation such as performance requirements, quality standards.

5) What is oops

Oops is basically 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.

6) Write Basic Concepts of oops

- Object
- Class
- Encapsulation
- Inheritance
- Polymorphism
  - Overriding
  - Overloading
- Abstraction

7) What is object

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.

Object is the basic unit of object oriented programming(OOP).

That is both data and function that operate on data are bundled as a unit called as object.

The two parts of an object

Object = Data + Methods or to say the same differently an object has the responsibility to know and the responsibility to do.

8) What is class

Class is defining as a blueprint for an object.

This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.

Class can be considered as the blueprint or definition or a template for an object and describes the properties and behavior of that object, but without any actual existence.

An object is a particular instance of a class which has actual existence and there can be many objects (or instances) for a class.

9) What is encapsulation

Encapsulation is placing the data and the functions that work on that data in the same place.

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.

Encapsulation enables data hiding**,** hiding irrelevant information from the users of a class and exposing only the relevant details required by the user.

10) What is inheritance

Inheritance means that one class inherits the characteristics of another class.

One of the most useful aspects of object-oriented programming is code reusability.

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 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 for classes and interfaces. Java supports multiple inheritances only through interfaces.

11) 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 behavior. A behavior is initiated by sending a message to the object.

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.

There are two types of polymorphism in Java

- Compile time polymorphism(Overloading)
- Runtime polymorphism(Overriding)

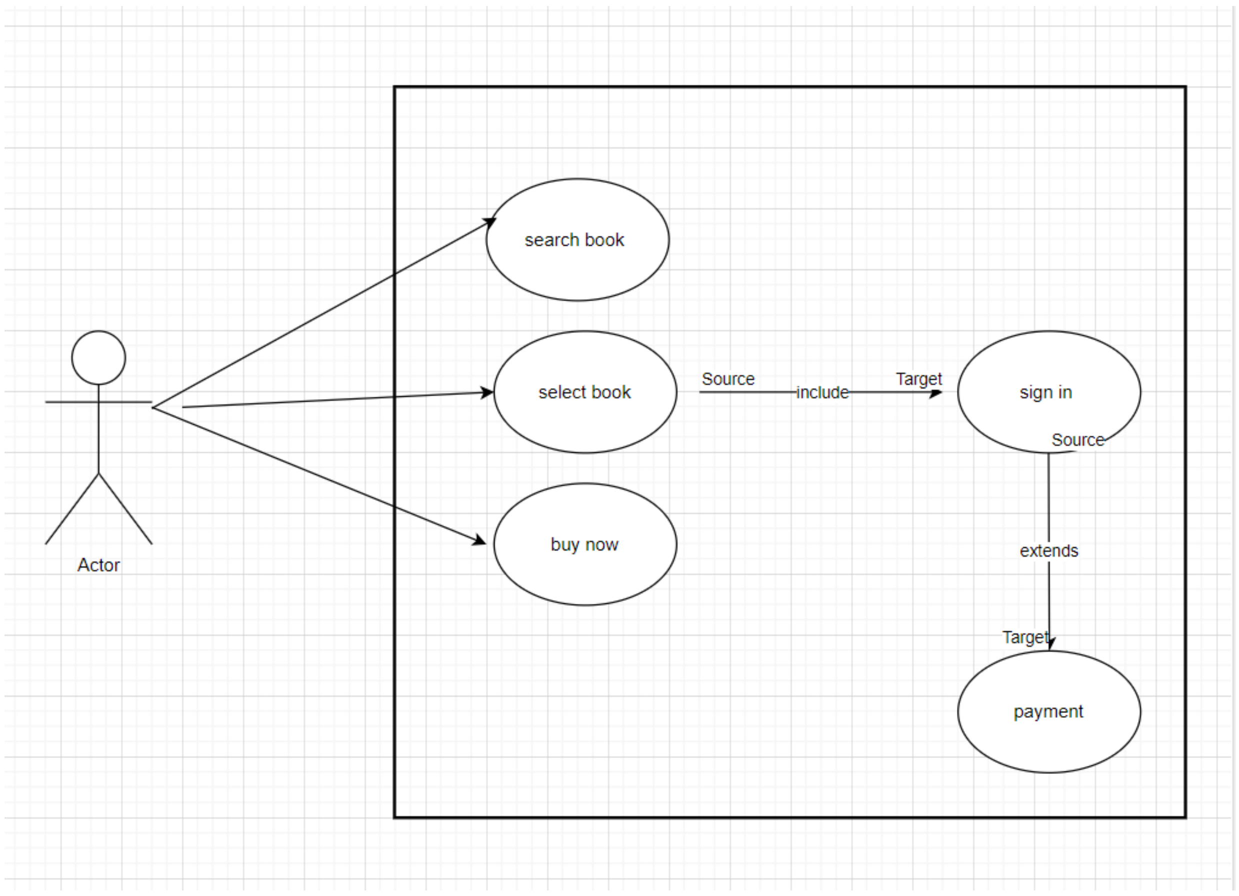
When the exiting operator or function is made to operate on new data type, it is said to be overloaded.

In method overloading, multiple methods having same name can appear in a class, but with different signature.

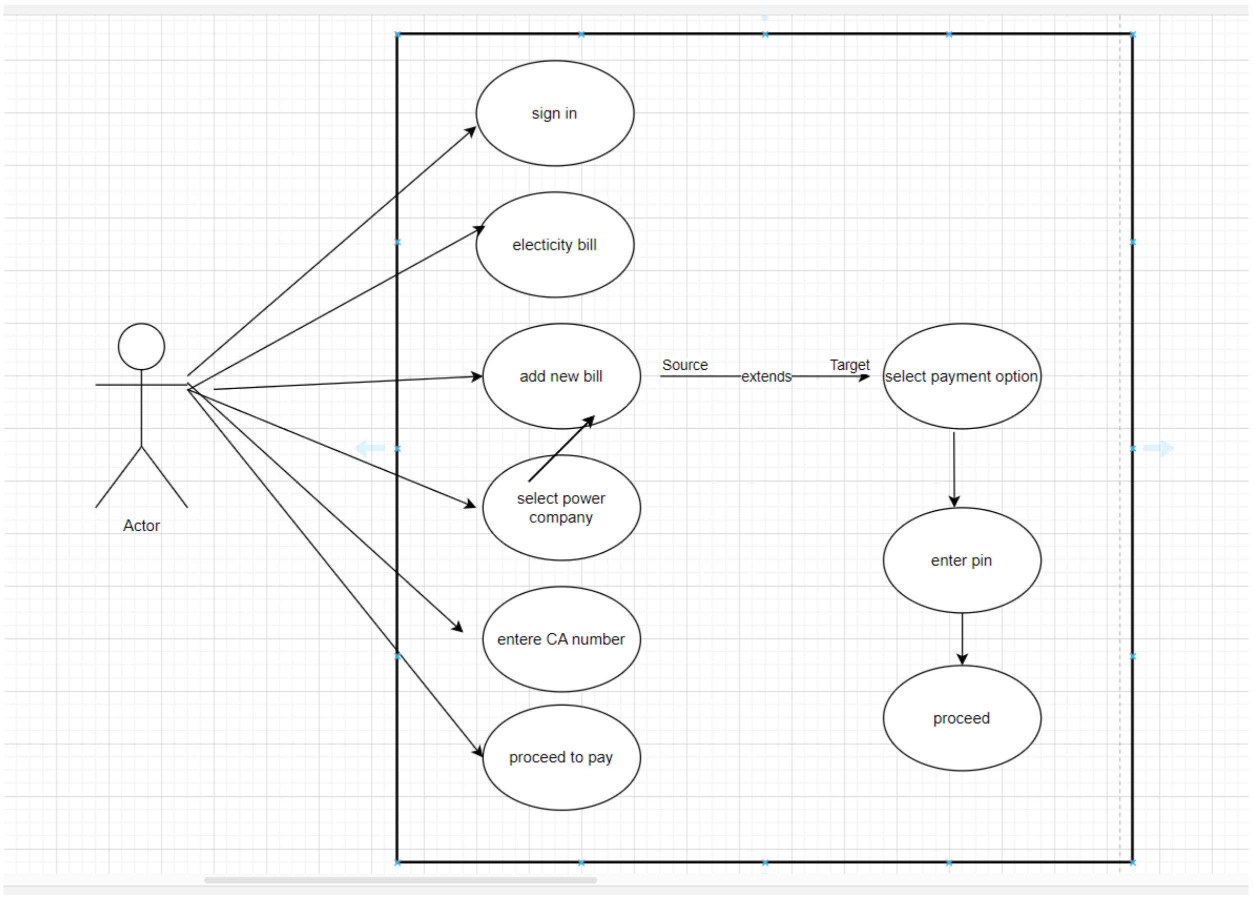
In other words, If a subclass provides the specific implementation of the method that has been declared by one of its parent class, it is known as method overriding.

Method overriding is used for runtime polymorphism

12) Draw Usecase on Online book shopping



13) Draw Usecase on online bill payment system (paytm)



14) Write SDLC phases with basic introduction

- Requirements Collection/Gathering -> Establish Customer Needs

Requirements definitions usually consist of natural language, supplemented by (e.g., UML) diagrams and tables.

Types of Requirements:

Functional Requirements: describe system services or functions.

Non-Functional Requirements: are constraints on the system or the development process.
Non-functional requirements may be more critical than functional requirements.

- Analysis -> Model and Specify the requirements- “What”

This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture.
The architecture defines the components, their interfaces and behaviors.
Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established.
The design may include the usage of existing components.

- Design -> Model And Specify a Solution – “Why”

Design Architecture Document

- Implementation Plan
- Critical Priority Analysis
- Performance Analysis
- Test Plan

The Design team can now expand upon the information established in the requirement document.

- Implementation -> Construct a Solution in Software

In the implementation phase, the team builds the components either from scratch or by composition.

Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.

- Testing -> Validate the solution against the requirements

The testing phase is a separate phase which is performed by a different team after the implementation is completed.

There is merit in this approach; it is hard to see one’s own mistakes, and a fresh eye can discover obvious errors much faster than the person who has read and re-read the material many times.

- Maintenance -> Repair defects and adapt the solution to the new requirements

Maintenance is the process of changing a system after it has been deployed.

Corrective maintenance: identifying and repairing defects

Adaptive maintenance: adapting the existing solution to the new platforms.

Perfective Maintenance: implementing the new requirements In a spiral lifecycle.
everything after the delivery and deployment of the first prototype can be considered “maintenance”.

15) Explain Phases of the waterfall model

Requirements: The first phase involves gathering requirements from stakeholders and analyzing them to understand the scope and objectives of the project.

Design: Once the requirements are understood, the design phase begins. This involves creating a detailed design document that outlines the software architecture, user interface, and system components.

Implementation: Implementation involves coding the software based on the design specifications. This phase also includes unit testing to ensure that each component of the software is working as expected.

Testing: In the testing phase, the software is tested as a whole to ensure that it meets the requirements and is free from defects.

Deployment: Once the software has been tested and approved, it is deployed to the production environment.

Maintenance: The final phase of the Waterfall Model is maintenance, which involves fixing any issues that arise after the software has been deployed and ensuring that it continues to meet the requirements over time.

16) Write phases of spiral model

The Spiral Model is a risk-driven model, meaning that the focus is on managing risk through multiple iterations of the software development process. It consists of the following phases:

1. **Objectives Defined:** In first phase of the spiral model we clarify what the project aims to achieve, including functional and non-functional requirements.
2. **Risk Analysis:** In the risk analysis phase, the risks associated with the project are identified and evaluated.

3. **Engineering:** In the engineering phase, the software is developed based on the requirements gathered in the previous iteration.

4. **Evaluation:** In the evaluation phase, the software is evaluated to determine if it meets the customer’s requirements and if it is of high quality.

5. **Planning:** The next iteration of the spiral begins with a new planning phase, based on the results of the evaluation.

17) Write agile manifesto principles

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration.

Each build is incremental in terms of features; the final build holds all the features required by the customer.

18) Explain working methodology of agile model and also write pros and cons.

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

Iterative approach is taken and working software build is delivered after each iteration
Each build is incremental in terms of features; the final build holds all the features required by the customer.

- ✓ Pros

  - Is a very realistic approach to software development.

  - Promotes teamwork and cross training.

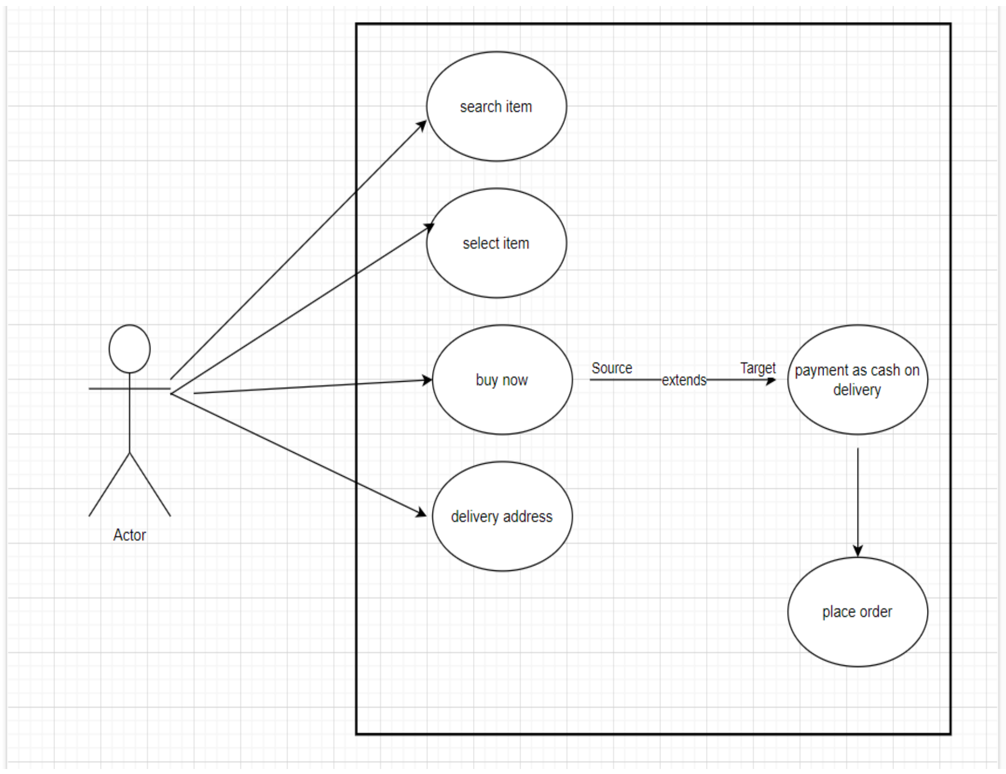
  - Functionality can be developed rapidly and demonstrated.

- Suitable for fixed or changing requirements.
- Delivers early partial working solutions.
- Easy to manage

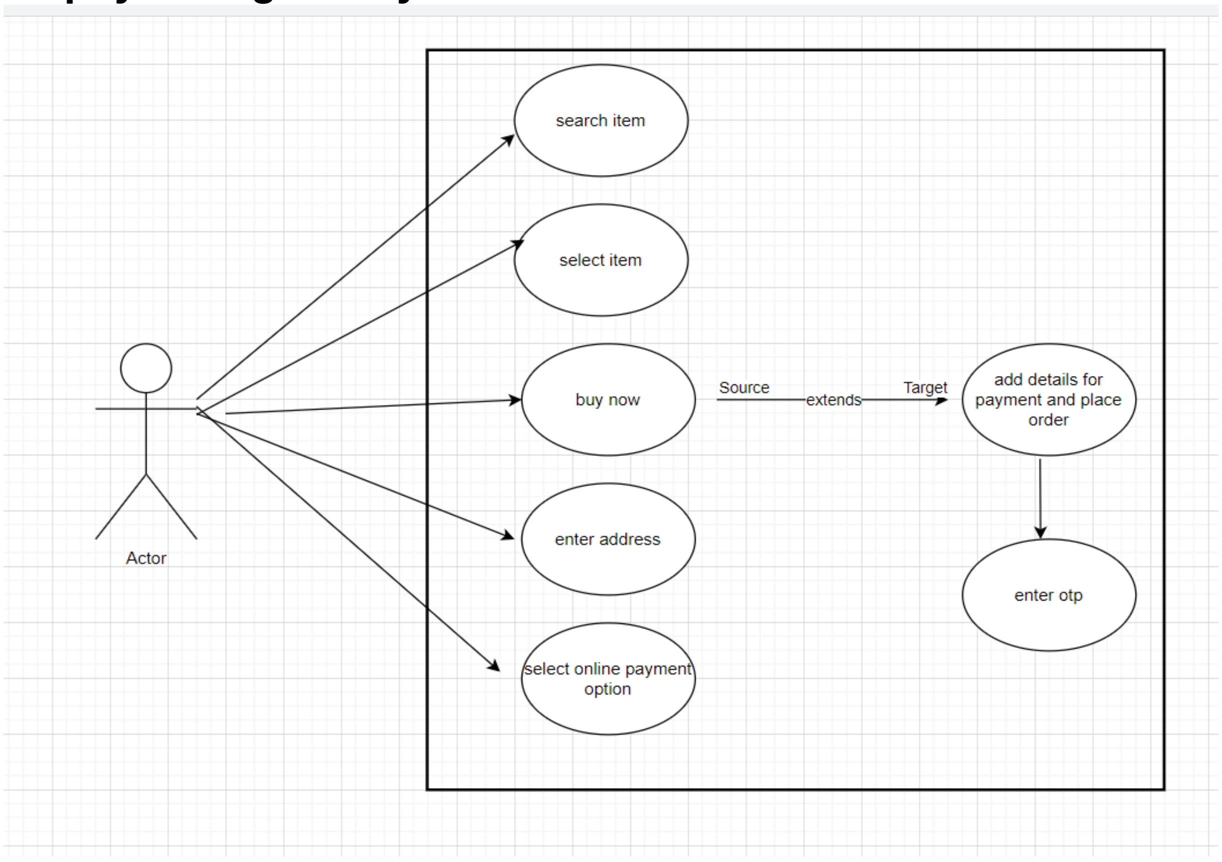
✓ Cons

- 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.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.

19) Draw usecase on Online shopping product using COD.



20) Draw usecase on Online shopping product using payment gateway.


search item
select item
buy now
Source
extends
Target
add details for payment and place order
enter address
enter otp
select online payment option
Actor