What is SDLC?

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. • A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and lifecycle management of an application or piece of software. • The methodology within the SDLC process can vary across industries and organizations, but standards such as

ISO/IEC12207 represent processes that establish a life cycle for software, and provide a mode for the development, acquisition, and configuration of software systems.

What is software testing?

**Software Testing** is a method to check whether the actual software product matches expected requirements and to ensure that software product is[Defect](https://www.guru99.com/defect-management-process.html)free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements.

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.

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 an iso contains nonfunctional (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). • Recommended approaches for the specification of software requirements are described by IEEE830-1998. • This standard describes possible structures, desirable contents, and qualities of a software requirements specification.

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 datatype.

• Object-oriented programming has a way of interacting objects, each house keeping its own state. • Objects of a program interact by sending messages to each other. Everything in the world is an object • A flower, a tree, an animal • A student, a professor • A desk, a chair, a classroom, a building • A university, a city, a country • The world, the universe • As subject such as CS, IS, Math, History,

Concepts of OOPS: • Object • Class • Encapsulation • Inheritance • Polymorphism • Abstraction

Write Basic Concepts of oops?

What is an object?

• Tangible Things • Roles • Incidents • Interactions • Specifications So, what are 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 concept applies. • This 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. • Object =Data + Methods • To say the same differently • An object has the responsibility to know and the responsibility to do.

What is class

Class: • When you define a class, you define 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. • A class represents an abstraction of the object and abstracts the properties and behavior of that object. • Class can be considered as the blueprint ordination 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. • In the case of a car or laptop, there will be a blueprint or design created first and then the actual car or laptop will be built based on that. • We do not actually buy these blueprints but the actual objects. • The two steps of Object • Oriented Programming • Making Classes: Creating, extending or reusing abstract datatypes. • Making Objects interact: Creating objects from abstract data • types and defining their relationships.

What is encapsulation

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 been closed in or as If in a capsule. In Java, a class is the capsule (or unit). Encapsulation • 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. • We can expose our operations hiding the details of what is needed to perform that operation. • We can protect the internal state of an object by hiding its attributes from the outside world (by making it private), and then exposing them through setter and getter methods. Now modifications to the object internals are only controlled through these methods.

What is abstraction

Abstraction: •

Abstraction is the representation of the essential features of an object. • These are encapsulated into an abstract data type. • Data abstraction refers to, providing only essential information to the outside word and hiding their background details, i.e., to represent the needed information in program without presenting the details. • For example, a database system hides certain details of how data is stored and created and maintained. • Similar way, C++ classes provides different methods to the outside world without giving internal detail about those methods and data. • In plain English, abstract means a concept to ride a not associated with Any specific instance and does not have a concrete existence.

Abstraction • Abstraction in Object Oriented Programming refers to the ability to make a class abstract. • Abstraction captures only those details about an object that are relevant to the current perspective. • Abstraction tries to reduce and factor out details so that the programmer can focus on a few concepts at a time. • Java provides interfaces and abstract classes for describing abstract types. • An interface is a contract or specification without any implementation. • An interface can't have behavior or state. • An abstract class is a class that cannot be instantiated. All other functionality of the class still exists. • Abstract classes can have state and can be used to provide a skeletal implementation. • A detailed comparison of interfaces and abstract classes can be found at interface v/s abstract-class.

What is polymorphism

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 (the things it can do). • A behavior 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. • Poly refers too many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism. • Many ways different upon the usage is called polymorphism. • E.g., the message display Details () of the Person class should give different results when send to a student object (e.g., the enrolment number). • The ability to change form is known as polymorphism. • There are two types of polymorphism in Java • Compile time polymorphism (Overloading)Runtime polymorphism (Overriding) Overloading • The concept of overloading is also a branch of polymorphism. When the exiting operator or function is made to operate on new datatype, it is said to be overloaded. • The same method name (method overloading) or operator symbol (operator overloading) can be used in different contents. • In method overloading, multiple methods having same name can appear in a class, but with different signature. • And based on the number and type of arguments we provide while calling the method, the correct method will be called. • Java doesn't allow operator overloading yet + is overloaded for class String. The ‘+’ operator can be used for addition as well as string concatenation

What is inheritance

Inheritance •

Inheritance means that one class inherits the characteristics of another class. This is also called a “isa” relationship. • 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 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 multi-level 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 a simple mentation inheritance, and in an interface context, it is also referred to as interface inheritance.

Write SDLC phases with basic introduction

Software Development Life Cycle • 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. • A Software Development Life Cycle is essentially a series of steps, or phases, that provide a model for the development and life cycle management of an application or piece of software. • The methodology within the SDLC process can vary across industries and organizations, but standards such as ISO/IEC12207 represent processes that establish a life cycle for software, and provide a mode for the development, acquisition, and configuration of software systems.

Explain Phases of the waterfall model

The sequential phases in Waterfall model are −

* **Requirement Gathering and analysis** − All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
* **System Design** − The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
* **Implementation** − With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
* **Integration and Testing** − All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
* **Deployment of system** − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
* **Maintenance** − There are some issues which come up in the client environment. To fix those issues, patches are released. Also, to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.



Write phases of spiral model

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e., the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

## Spiral Model - Design

The spiral model has four phases. A software project repeatedly passes through these phases in iterations called Spirals.

### **Identification**

This phase starts with gathering the business requirements in the baseline spiral. In the subsequent spirals as the product matures, identification of system requirements, subsystem requirements and unit requirements are all done in this phase.

This phase also includes understanding the system requirements by continuous communication between the customer and the system analyst. At the end of the spiral, the product is deployed in the identified market.

### **Design**

The Design phase starts with the conceptual design in the baseline spiral and involves architectural design, logical design of modules, physical product design and the final design in the subsequent spirals.

### **Construct or Build**

The Construct phase refers to production of the actual software product at every spiral. In the baseline spiral, when the product is just thought of and the design is being developed a POC (Proof of Concept) is developed in this phase to get customer feedback.

Then in the subsequent spirals with higher clarity on requirements and design details a working model of the software called build is produced with a version number. These builds are sent to the customer for feedback.

### **Evaluation and Risk Analysis**

Risk Analysis includes identifying, estimating and monitoring the technical feasibility and management risks, such as schedule slippage and cost overrun. After testing the build, at the end of first iteration, the customer evaluates the software and provides feedback.

The following illustration is a representation of the Spiral Model, listing the activities in each phase.



Based on the customer evaluation, the software development process enters the next iteration and subsequently follows the linear approach to implement the feedback suggested by the customer. The process of iterations along the spiral continues throughout the life of the software.

Write agile manifesto principles

Agile Model/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 stake holders.

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 suited for 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. • Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

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. • Suitable for fixed or changing requirements • Delivers early partial working solutions. • 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

Cons: • Not suitable for handling complex dependencies. • More risk of sustainability, maintain ability 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. • Transfer of technology to new team members maybe quite challenging due to lack of documentation