**Software Testing Assignment**

**Module-1**

1. What is SDLC?

**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 lifecycle management of an application or piece of software.

**SDLC Phases**

|  |  |
| --- | --- |
| **Requirements**  **Collection/Gathering** | **Establish Customer Needs** |
| **Analysis** | **Model And Specify the requirements-**  **“What”** |
| **Design** | **Model And Specify a Solution – “Why”** |
| **Implementation** | **Construct a Solution In Software** |
| **Testing** | **Validate the solution against the**  **Requirements** |
| **Maintenance** | **Repair defects and adapt the solution to the new requirements** |

**2. What is software testing?**

* Testing is the process of evaluating a system or its component with the intent to find that weather it satisfies the specified requirements or not.
* This activity results in the actual, expected and difference between their results.
* In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.
* Software testing is the process used to identify the correctness, Completeness, and quality of developed computer software.

**3. What is agile methodology?**

**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 stakeholders.

**What is Agile?**

* 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.
* 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, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is very high individual dependency, since there is minimum documentation generated.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.

**4. What is SRS?**

**Software Requirement Specification**

* 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 (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).

**5. What is oops?**

* Identifying objects and assigning responsibilities to these objects.
* Objects communicate to other objects by sending messages.
* Messages are received by the methods of an object

An object is like a black box.

* The internal details are hidden.
* Object is derived from abstract data type
* Object-oriented programming has a web of interacting objects, each house-keeping its own state.
* Objects of a program interact by sending messages to each other.

**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
* A subject such as CS, IS, Math, History, ...

**6. Write basic Concepts of oops?**

Concepts of OOP

* 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. 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.

**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?**

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 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. 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.

**9. What is encapsulation?**

* Encapsulation is the practice of including in an object everything it needs hidden from other objects. The internal state is usually not accessible by other objects.
* Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object- oriented programming provides you framework to place the data and the relevant functions together in the same object.
* Encapsulation in Java is the process of wrapping up of data (properties) and behavior (methods) of an object into a single unit; and the unit here is a Class (or interface).
* Encapsulate in plain English means to enclose or be enclosed in or as if in a capsule. In Java, a class is the capsule (or unit).

**10. What is inheritance?**

* Inheritance means that one class inherits the characteristics of another class. This is also called a “is a” relationship
* One of the most useful aspects of object-oriented programming is code 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 multilevel inheritance (Grandparent-> Parent -> Child) for classes and interfaces. Java supports multiple inheritances (multiple parents, single child) only through interfaces.
* In a class context, inheritance is referred to as implementation inheritance, and in an interface context, it is also referred to as interface inheritance.

|  |
| --- |
| parent |

|  |
| --- |
| Grandparent |

|  |
| --- |
| child |

**11. What is polymorphism?**

* Polymorphism means “having many forms”.
* It allows different objects to respond to the same message in different

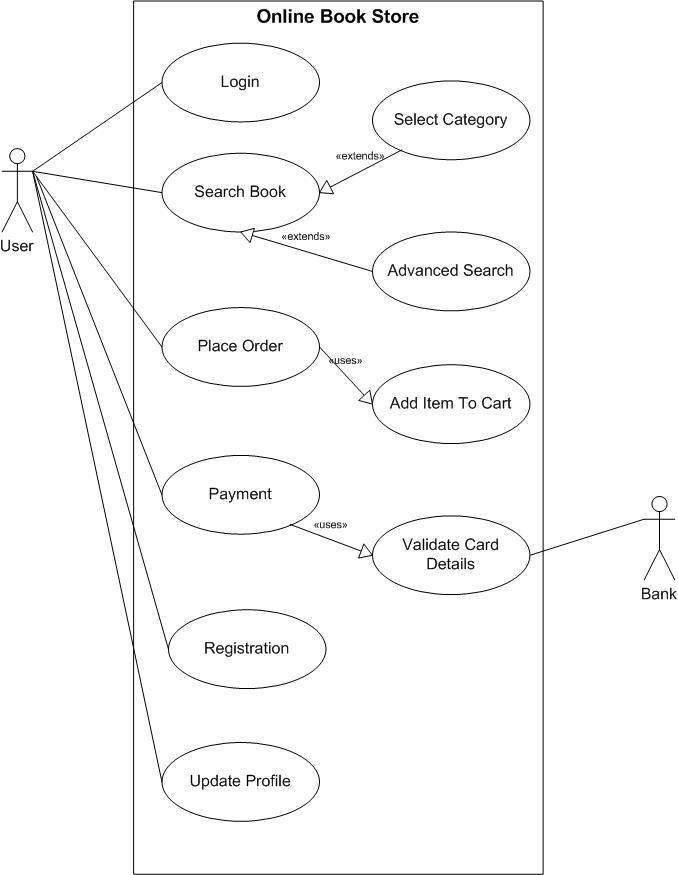
ways, the response specific to the type of the object.

* The most important aspect of an object is its behaviour (the things it can do).
* A behaviour is initiated by sending a message to the object (usually by calling a method).
* The ability to use an operator or function in different ways in other words
* giving different meaning or functions to the operators or functions is called polymorphism.
* Poly refers to many. That is a single function or an operator functioning in
* many ways different upon the usage is called polymorphism.
* E.g. the message displayDetails() 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.

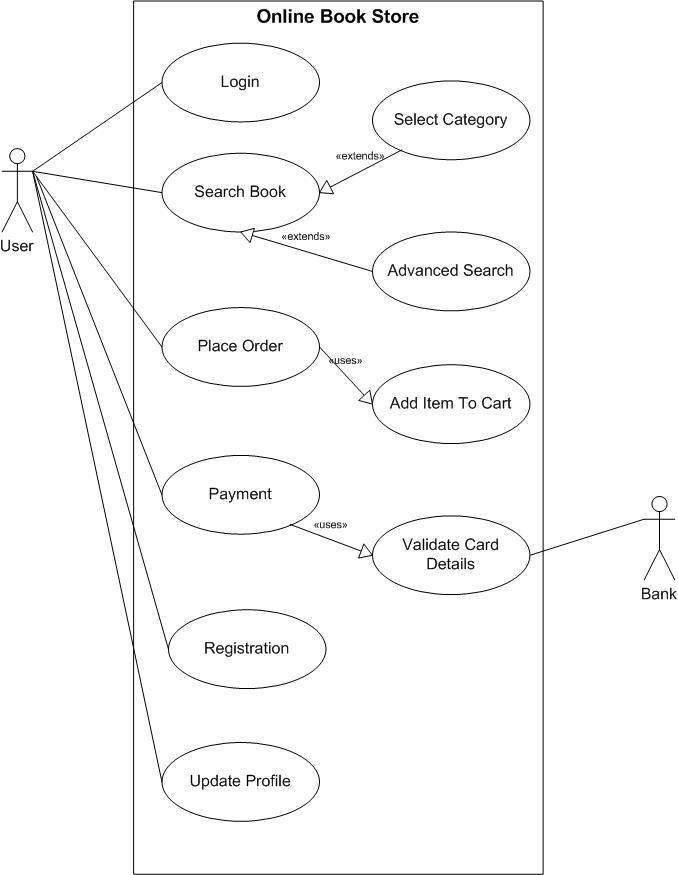
There is two types of polymorphism in Java

1. Compile time polymorphism(Overloading)
2. Runtime polymorphism(Overriding)

**12. Draw usecase on online book shopping?**

****

**13. Draw usecase on online bill payment system (paytm)?**

****

**14. Write SDLC phases with basic introduction?**

**The classical software lifecycle models the software development as a step-by-step <waterfall> between the various development phases.**

The waterfall is unrealistic for many reasons, especially:

* Requirements must be “frozen” to early in the life cycle
* Requirements are validated too late

**Applications(When to use?)**

* Requirements are very well documented, clear and fixed.
* Product definition is stable.
* Technology is understood and is not dynamic.
* There are no ambiguous requirements.
* Ample resources with required expertise are available to support the product.
* The project is shor**t.**

**15. Explain phases of the waterfall model?**

**Pros (Why Waterfall Model)**

* Simple and easy to understand and use
* Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
* Phases are processed and completed one at a time.
* Works well for smaller projects where requirements are very well understood.
* Clearly defined stages.
* Well understood milestones.
* Easy to arrange tasks.
* Process and results are well documented.

**Cons (Why not Waterfall Model)**

* No working software is produced until late during the life cycle.
* High amounts of risk and uncertainty.
* Not a good model for complex and object-oriented projects.
* Poor model for long and ongoing projects.
* Not suitable for the projects where requirements are at a moderate to
* high risk of changing. So risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* No working software is produced until late in the life cycle.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.

**16. Write phases of spiral model?**

1. Planning: determination of objectives, alternatives and constraints.

2 Risk analysis : analysis of alternatives and identification\resolution of risks.

3. Engineering: development of the <next level> of product.

1. Customer evaluation: assessment of the result of engineering.

**Application**

* Spiral Model is very widely used in the software industry as it is in synch with the natural development process of any product i.e. learning with
* maturity and also involves minimum risk for the customer as well as the development firms. Following are the typical uses of Spiral model:
* When costs there are a budget constraint and risk evaluation is important. For medium to high-risk projects.
* Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
* Customer is not sure of their requirements which are usually the case.
* Requirements are complex and need evaluation to get clarity.
* New product line which should be released in phases to get enough customer feedback.
* Significant changes are expected in the product during the development cycle.

**Pros (Why It works)**

* Changing requirements can be accommodated.
* Allows for extensive use of prototypes
* Requirements can be captured more accurately.
* Users see the system early.
* Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

**Cons (Why It doesn’t work)**

* Management is more complex.
* End of project may not be known early.
* Not suitable for small or low risk projects and could be expensive for small projects.
* Process is complex
* Spiral may go indefinitely.
* Large number of intermediate stages requires
* excessive documentation.

**17. Write agile manifesto principals?**

* 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.

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

**What is Agile?**

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

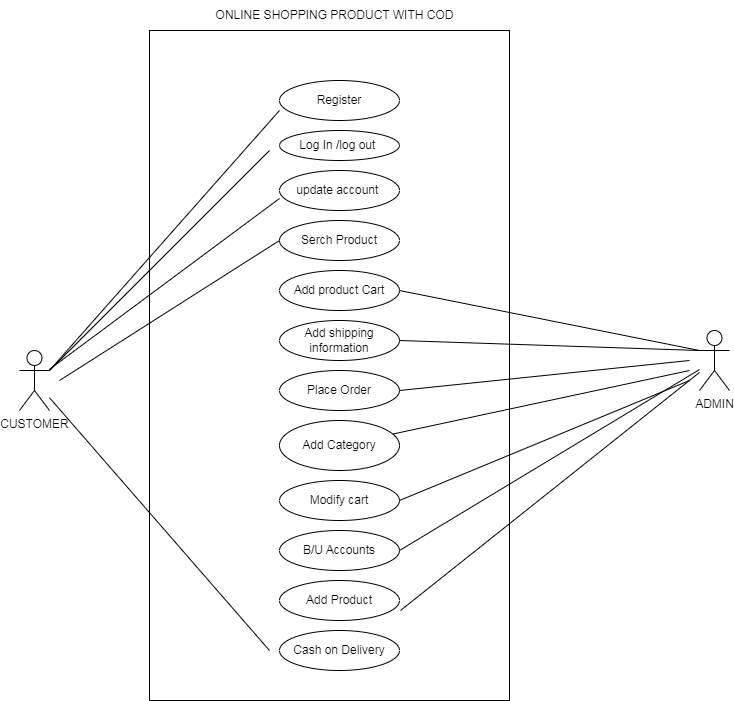
**Pros**

* 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

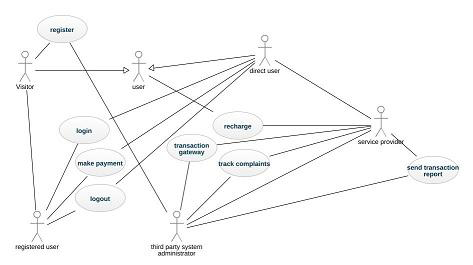
**Cons**

* Not suitable for handling complex dependencies.
* More risk of sustainability, maintainability and extensibility.
* An overall plan, an agile leader and agile PM practice is a must without which it will not work.
* Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
* Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
* There is very high individual dependency, since there is minimum documentation generated.
* Transfer of technology to new team members may be quite challenging due to lack of documentation.

**19. Draw usecase on online shopping product using COD?**

****

**20. Draw usecase on online shopping product using payment gateway?**

****