**PRACTICAL – 5**

**AIM**: List at least 10 software design principles & online/offline tools for software development process and Draw the UML diagram for Microsoft Windows 10 operating system.

**THEORY** Here we will be discussing about a number of software design principles and also some of the tools for software development process.

**Software Design Principles:-** The first five principles form the basics of the software development principles and are also known as “S.O.L.I.D principles”.

1. **Single Responsibility Principle (SRP):-** It is a software engineering principle that states that a class should have only one reason to change. In other words, it must have only one responsibility.Here, we are talking about cohesion. All elements in given class structures or modules should have a functional affinity to one another. By clearly defining your class’s responsibility, you increase its cohesiveness. This is the first principle denoted by **“S”.**
2. **Open/Closed Principle (OCP):-** The principle says that you should be able to change the behavior of a class without modifying it. Therefore, you can extend the class’s behavior through composition, interface, and inheritance. However, you cannot open it for minor modifications. This is the second principle denoted by **“O”.**
3. **Liskov Substitution Principle (LSP):-**The LSP principle mainly focusses on the degree of use of the Inheritance in a software. While inheritance is beneficial, it is advisable to use it contextually and moderately. The principle strives to prevent cases where classes are extended only through common things. You need to consider the pre-conditions and post-conditions of a class before performing inheritance. This is the third principle denoted by **“L”.**
4. **Interface Segregation Principle (ISP):-** ISP prefers many specific interfaces to a general interface. The goal is to have finely grained and client-specific interfaces. You need to enhance cohesion in interfaces and develop modules with few behaviors. Interfaces that have many behaviors are hard to maintain and evolve. So, they should be completely avoided. This is the fourth principle denoted by **“I”.**
5. **Dependency Inversion Principle (DIP):-** The fifth and the final principle of SOLID is Dependency Inversion Principle and is denoted by **“D”.** The principle asserts that programmers should depend on abstractions and not on concrete classes. This can be broken down to 2 parts as: We can break it into two:

* High-level modules need to be independent of low-level ones. Both should depend on abstractions
* Abstractions should be independent of details. Details should depend on abstractions.

Now we look at some of the important principles other than SOLID .

1. **Keep It Simple :-** Ensuring that the program coding is simple and very easy to understand is very much important. The code shouldn’t give a new person hard time to figure out how things are working. Methods and functions should be as brief as possible but also easily understandable. Each of them should solve only one problem or two at max. Also the project code should not have a lot of conditions (simple and nested conditions). Optimizing the conditions would help understanding and finding bugs easily for the QA team.
2. **You Aren't Gonna Need It (YAGNI):-** Most programmers fall into the pit of trying to implement all the functionalities at once, from the word go. In the end, some or most of these functionalities become useless. Start by adding just a few methods to a class. After that, as your project starts taking shape and new demands arise, you can add more functionalities. That way, you’ll achieve a lean development software. YAGNI saves time, efforts, and costs that you would have wasted in trying to understand or debug the code.
3. **Measure Twice and Cut Once:-** The development life cycle’s requirement stage usually introduces more than 50% coding issues if not done well. Therefore a systematic approach should be developed. It is very much important to double check all the project requirements so that one does not miss any important section(feature) OR does not accidentally add too much. After that, make blueprints that will guide the whole process to achieve high-quality coding throughout. Always test your project from basics to ensure everything is fine. This principle gives much more predictable outcomes, especially if the project’s cost is already high. You’ll save yourself headaches that come with deleting or adding code lines to meet requirements.
4. **Don’t Repeat Yourself (DRY):-** When writing your code, don’t repeat yourself. That is, avoid copy-pasting your code in different places. Otherwise, future maintenance will be difficult. The reason is that you will have to make changes to the coding in those various places. Those changes will further necessitate changes in the tests to make the results click green. All of that will need more time, effort, and money. To avoid such a pitfall, you can extract a common logic into functions. Additionally, if there are any manual works that you can automate, do so to keep your code lean. For software development, the above steps will help in the code re-usability without having to repeat it.
5. **Least Astonishment:-** The principle of least astonishment says that it is advisable to design a feature that doesn’t have a high-astonishment factor. Your system’s components should behave in a way that end-users expect. Therefore, your project’s outcomes will be profitable only if they are obvious, predictable, and consistent. Otherwise, users will shy from using features or structures that astonish, surprise, or confuse them. You are making software products for people to use. Thus, you’ll reap a lot by designing user-friendly features. Strive to match human beings’ mental models, experience, and expectations. Remember, you have to capture the user’s attention as quickly as possible. As we know, the current users’ attention span has plummeted.

**SOFTWARE DEVELOPMENT PROCESS TOOLS:**

1. **Embold:-** Fixing bugs before deployment saves a lot of time and energy in the long run. Embold is a software analytics platform that analyses source code and uncovers issues that impact stability, robustness, security, and maintainability.

**Features:**

* With the Embold plugins, you can pick up code smells and vulnerabilities as you code, before making commits.
* Unique anti-pattern detection prevents the compounding of unmaintainable code.
* Integrate seamlessly with Github, Bitbucket, Azure, and Git and plugins available for Eclipse and IntelliJ IDEA.
* Get deeper and faster checks than standard code editors, for over 10 languages.

1. **Linx:** Linx is a low code IDE and server. IT pros use Linx to quickly create custom automated business processes, integrate applications, expose web services and to efficiently handle high workloads.

**Features:-**

* Easy-to-use, drag-and-drop interface
* Over 100 pre-built functions and services for rapid development
* One-click deployment to any local or remote Linx Server directly from the IDE
* Input and outputs include nearly any SQL & NoSQL databases, numerous file formats (text and binary) or REST and SOAP Web services
* Live debugging with step through logic
* Automate backend processes via timer, directory events or message queue or expose web services, and call APIs via HTTP requests

1. **Studio 3T:-** Studio 3T for MongoDB helps you to build queries fast, generate instant code, import/export in multiple formats, and much more.

**Features:-**

* Query MongoDB faster with our Visual Query Builder, IntelliShell, or SQL Query tool.
* Our Data Masking tool enables data compliance and bolsters security with powerful field-level data obfuscation.
* Import to MongoDB from JSON, CSV, BSON/mongodump, and SQL, and get a preview of your output documents as you make changes.
* Migration from MongoDB to SQL (or vice versa) has never been easier with our Migration tools."

1. **Kite:-** Kite is IDE for Software Development that automatically completes multiple line codes. This editor supports more than 16 languages. It helps you to code faster with no hassle.

**Features:-**

* It offers Software Development documentation.
* This editor provides a function signature as you type.
* You will get a tooltip on mouse hover.
* Provides support in email.
* Uses machine learning models for Software Development language.
* Also it is a free to use open source tool.

1. **NetBeans:-** NetBeans is an open source and a free software development tool written in Java that develops world-class web, mobile, and desktop applications easily and quickly. It uses C / C++, PHP, JavaScript, Java etc.

**Features:-**

* Support for fast & smart code editing.
* Easy & Efficient Project Management process.
* Rapid User Interface Development.
* Helps to write bug-free code.
* NetBeans IDE offers superior support for C/C++ and PHP developers.

1. **Cloud9 IDE:-** Cloud9 IDE is an online integrated software development environment. It supports many programming languages like C, C++, PHP, Ruby, Perl, Python, JavaScript and Node.js.

**Features:-**

* Allows to clone entire development environment.
* Built-In Terminal for command-line wizard.
* Code Completion suggestions helps software developers to code faster and avoid typos.
* The Debugger helps developers to set breakpoints, and inspect variables of any JS/Node.js app.
* Simply drag any file or Terminal to create multiple split views.
* Developers can select an extensive set of default Runners to execute app, such as Ruby, Python, PHP/Apache.

1. **Atom:-**Atom is a solid all-around text-editor. It is fully free and open source. It can be customized to do anything but without a need of modifying the config file.

**Features:-**

* Atom works across many popular operating systems like OS X, Windows, or Linux.
* It helps developers to write code faster with a smart, flexible autocomplete.
* Easily browse and open whole project or multiple projects in one window.
* It is possible to split Atom interface into multiple panes to compare and edit code across files.
* Find, preview, and replace text type in a file or across the entire project.

1. **GitHub:-** GitHub is a powerful collaboration tool and development platform for code review and code management. With this GitHub, the users can build applications and software, manage the projects, host the code, review the code etc.

**Features:-**

* With GitHub, developers can easily document their code and can host the same from the repositories.
* GitHub’s project management tools help its users to stay aligned, co-ordinate easily and get their task done accordingly.
* Few features of GitHub that make it a useful tool are its code security, access control among the team members, integration with other tools etc.
* Few developers use GitHub for experimenting new programming languages in their personal projects.
* GitHub can be hosted on servers and on a cloud platform. It runs on Windows and Mac OS.
* GitHub is free for open source projects and public use. For developers it is charged based on different criteria and services requested.

1. **Bitbucket:-** Bitbucket is a distributed, web-based version control system that is used for collaboration between software development teams (code and code review). It is used as a repository for source code and development projects.

**Features:-**

* Useful features of Bitbucket that makes it a powerful tool are its flexible deployment models, unlimited private repositories, code collaboration on steroids etc.
* Bitbucket supports few services like code search, issue tracking, Git large file storage, bitbucket pipelines, integrations, smart mirroring etc.
* Using Bitbucket, one can organize the repositories into the projects with which they can focus easily on their goal, process or product.
* To rationalize the development process of any software it can integrate into the prevailing workflow.
* Bitbucket offers a free plan for 5 users with unlimited private repositories.

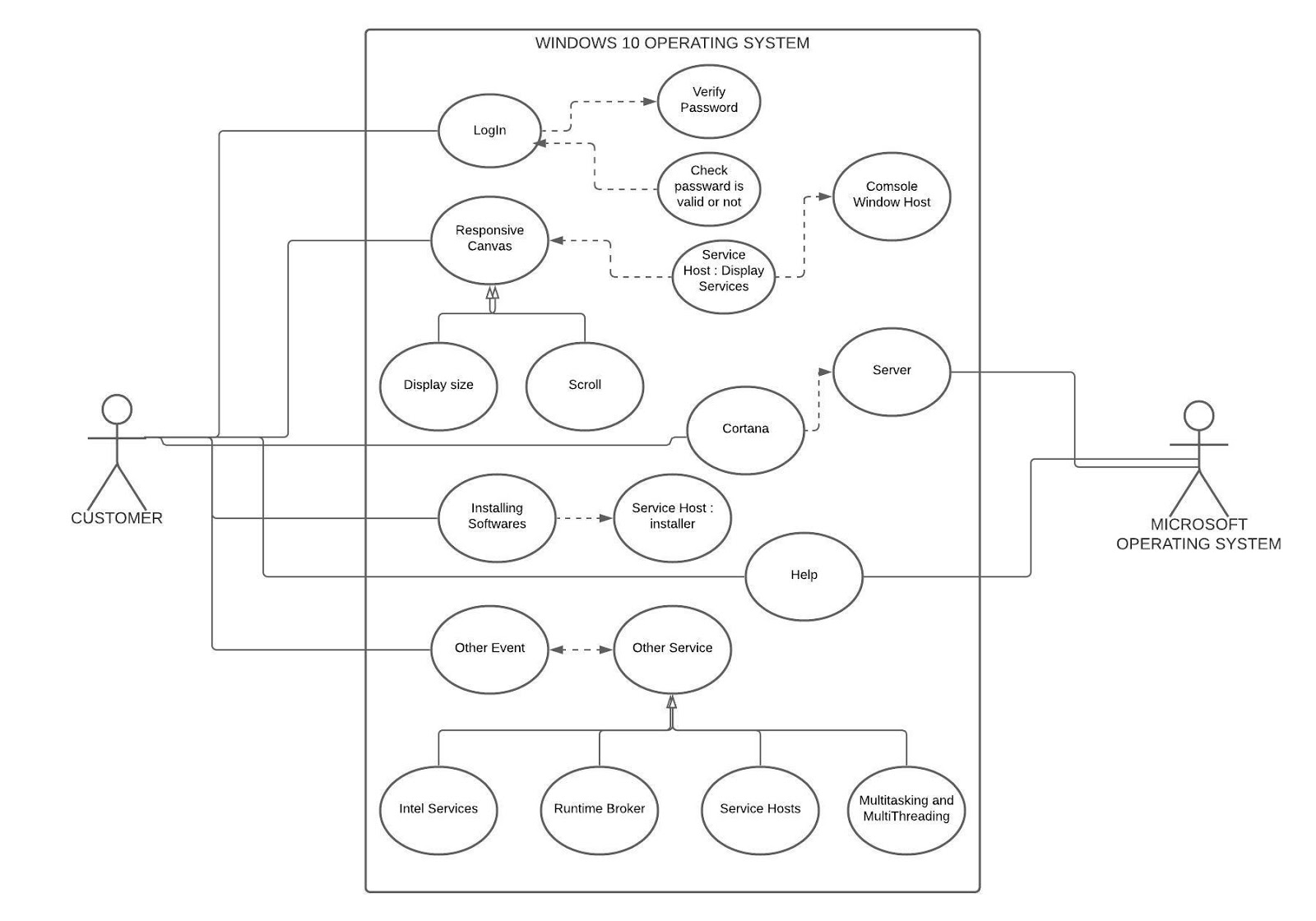
1. **JIRA:-**Jira is the most popular software development tool that is used by agile teams for planning, tracking and releasing the software.

**Features:**-

* This tool is customizable and also has some prevailing features that are used in every development phase.
* Using Jira, we can accomplish the work in progress, generate reports, backlogs etc.
* Few other important features of Jira software are Scrum boards, Kanban boards, GitHub integration, Disaster recovery, Code Integration, Portfolio Management, Sprint Planning, Project Management etc.
* Jira works for Windows and Linux/Solaris operating systems.

**UML DIAGRAM FOR WINDOWS 10 OPERATING SYSTEM**

UML, short for Unified Modeling Language, is a standardized modeling language consisting of an integrated set of diagrams, developed to help system and software developers for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML uses mostly graphical notations to express the design of software projects. Using the UML helps project teams communicate, explore potential designs, and validate the architectural design of the software. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing object oriented software and the software development process.



**CONCLUSION:**

In this practical, we learned about different software development principles to keep in mind during a software development planning. Also we have seen some of the most popularly used software development tools which are available online as well as application software. Also at the end we have learnt about UML diagrams, how to prepare them and also seen the UML diagram of windows 10 operating system.