Write short notes on the following:

**Lean Development**

Lean development is the use of Lean principles to software development. Lean principles originated in manufacturing to minimize waste and maximize value to the customer in the production line. These two goals are also useful in software development, which also follows an iterative process, requires quality standards and relies on the collaboration of a set of team members.

Lean Development is an [agile framework](https://www.productplan.com/glossary/agile-framework/) that is based on minimizing development time and efforts, eliminating waste, and delivering only the product requirements. It is often referred to as the [Minimum Viable Product (MVP)](https://www.productplan.com/resources-pp/mvp-truly-compelling/) strategy, in which a team:

1. Releases an MVP to the market

2. Learns from users what they like, don’t like and want to be added

3. Iterates based on this feedback

The seven Lean principles are:

* **Eliminate waste:** Lean philosophy regards everything not adding value to the customer as waste
* **Build quality in:** Efficient quality management using TDD, continuous integration, pair programming
* **Create knowledge:** learning is amplified through ample code reviewing, meetings and establishment of metrics
* **Defer commitment:** Decide as late as possible and schedule irreversible decisions for the last responsible moment
* **Deliver fast:** Deliver as fast as possible- The sooner the end product is delivered without major defects, the sooner feedback can be received
* **Respect people:** empowering team members to make important development decisions, based on knowledge they receive whilst writing code and their own judgment
* **Optimize the whole:** Lean software development focuses on optimising the value stream as a whole

When to use Lean Development?

It is successful when we require streamlining of the process and eliminating any activity not useful for the final product.

**Extreme Programming**

Extreme Programming is an Agile software development methodology that is primarily used to improve the quality of software and allow proper adaptation of the software to the changing needs of the client.

Extreme Programming aims to provide iterative small releases throughout the project, allowing the team members and clients to review and learn from the project’s progress throughout the entire [S](https://airbrake.io/blog/category/sdlc)oftware Development Life Cycle.

Extreme Programming rules are based on these simple **5 values**.

• **Simplicity**: Developers write simple code to add more value to a product, which in turn helps save time and efforts.

• **Communication**: Everyone on a team works jointly at every stage of the project, from gathering requirements to implementing code and getting feedback.

• **Respect**: Every person assigned to a project contributes to a common goal.

• **Feedback**: Team members deliver software frequently in order to get feedback about it, and make necessary improvements according to the new client requirements.

• **Courage**: Team members take full responsibility for their work and are always ready to respond to changes and suggest improvements.

**Common XP Practices:**

1. **Pair Programming**: This practice involves two programmers working jointly on the same code. It results in high quality software in spite of increased time required.
2. **Test Driven Development**: In this approach, each piece of code must pass the test to be released. This allows programmers to use immediate feedback and improve their code.
3. **Continuous Integration**: Team members commit code multiple times in a day, enabling continuous improvement and delivery.
4. **The Planning Game**: At the beginning of the cycle, the team members and the customer discuss product features and requirements.
5. **On-site Customer**: The end customer should fully participate in development to answer team questions and set priorities.
6. **Code Refactoring**: It involves removing redundancy, and eliminating unnecessary code.

When to use XP?

XP should be used when you expect experience constantly changing requirements or work with customers who aren’t sure what they want the system to do. Also, it is successful when you want to mitigate project risk, especially around tight deadlines.

**Adaptive Software Development**

Adaptive Software Development is an outgrowth of Rapid Software development. It focuses on rapid creation and development of softwares. It helps teams to quickly adapt to changing requirements or market needs by continuous planning and learning.

The process involves 3 phases:

1. Speculate

2. Collaborate

3. Learn

It replaces the traditional waterfall cycle with a repeating cycle of speculate, collaborate and learn. The characteristics of ASD is that it is mission focused, feature based, iterative, timeboxed, risk driven, and change tolerant.

The focus of adaptive software development is in the computer code. Instead of planning the software out beforehand, developers have a basic idea in their heads and they go to work. If there is any change or adaptation, the coders work on it based on their learnings.

The relatively short cycle of development leads to overall quick addition of new features and delivery of the end product.

When to use ASD?

ASD is a useful methodology when your organization prioritizes rapid delivery of products and sees value in continuously evolving the requirements. But if you do not support close involvement with your clients or continuous testing for your code, then it is not a viable solution.

**Feature Driven Development**

Feature Driven Development methodology is an Agile Development methodology that is customer-centric, iterative and incremental. It is known for short iterations and frequent releases of the software. It is a feature-focused method, i.e. It allows the project to be divided into features, that are small pieces of the complete project.

5 main activities of FDD:

1. **Develop a model**: The overall model is by identifying the [scope](https://searchcio.techtarget.com/definition/project-scope) of the system. Domain models are developed by the chief architect and other team members.
2. **Build a list of features**: A feature is a client valued output. Each feature is delivered within 2 weeks, therefore each feature should be manageable within that time.
3. **Plan out each feature**: Features are planned and assigned on the basis of how long they take to create and how important they are to the client.
4. **Design each feature**: The domain classes, sequence diagrams and general designs of each feature are inspected and finalized.
5. **Build each feature**: After the design is inspected and improved, the code is written, unit tested and the completed feature is added to the project.

When to use FDD?

FDD Approach is suitable for long term, complex projects. It is scalable and delivers fast results.