**Task -1:**

### **Good Code:**

* Clear and Readable: It’s easy to understand, even for someone who didn’t write it. You can look at the code and quickly figure out what it’s doing.
* Well-organized: The code is structured in a way that makes sense, using functions, loops, and variables to break down tasks clearly.
* Efficient: It gets the job done with minimal resources and in a reasonable amount of time.
* Easy to Maintain: If there’s a problem or new feature needs to be added later, it’s easy to make changes without breaking things.

**Bad Code:**

* Confusing: It’s hard to understand what the code is doing, even for someone who knows how to code. It might be messy or full of jargon.

In the bad code, it’s harder to see what’s going on, and there's no clear structure (e.g., no function for calculating the area). It works, but it’s not as readable or maintainable.

**Task -2:**

Data binding is a concept mostly used in frameworks and libraries for creating user interfaces, like React, Angular, Vue.js, or even in desktop/mobile development with frameworks like WPF or SwiftUI.

In simple terms, data binding is a way to link or connect your UI (user interface) elements to the underlying data or model. This means that whenever the data changes, the UI automatically updates to reflect those changes (and vice versa). It helps keep the interface and the underlying data in sync without needing to manually update the UI when the data changes.

**Task -3:**

Continuous development is a software practice where new features, improvements, and bug fixes are constantly created, tested, and delivered instead of waiting for big updates.

In simple words:

* Think of it like a bakery that makes fresh bread every day instead of once a month.
* Developers keep writing code and immediately check if it works.
* The changes are then automatically tested and, if all is good, quickly shared with users.

**Task -4:**

Polymorphism means "same action, different behavior" like how the word *"draw"* can mean drawing a picture for an artist or drawing a weapon for a soldier.

For polymorphism to work in programming (especially in Java), these conditions must be met:

1. Inheritance – There must be a parent (super) class and a child (sub) class.
2. Method Overriding – The child class must have a method with the same name, parameters, and return type as in the parent class.

Reference of Parent, Object of Child – You usually write something like:  
Parent obj = new Child();

1. This allows the method that runs to be decided at runtime.
2. Upcasting – The child object is treated as if it’s a parent type, but still behaves like the child when overridden methods are called.

Task -5:

## **1. TDD (Test-Driven Development)**

What is it?

A way of writing software where you write the tests first and then write the code to make those tests pass.

Why is it used?

To make sure your code works correctly from the start.

To catch bugs early before the program grows big.

Where is it used?

Used in software projects where quality and fewer bugs are important.

Common in backend systems, APIs, and critical applications.

## **2. BDD (Behavior-Driven Development)**

What is it?

An approach where you focus on how the software should behave from the user’s point of view, often using plain language

Tests are written in a way that non-programmers can read, like a story.

Why is it used?

To make sure developers, testers, and business people all understand the requirements.

Helps avoid misunderstandings between tech and non-tech people.

Where is it used?

In projects where communication between teams is critical.  
Common in web applications, UI testing, and collaborative projects.

**Task -6:**

### Manual Testing Tools (help testers but do not automate execution)

JIRA – Issue & bug tracking

TestLink – Test case managemen

Zephyr – Test management inside JIRA

qTest – Test case and defect tracking

PractiTest – Test case organization & reporting

Bugzilla – Bug tracking tool

MantisBT – Bug tracking and project management

### Automated Testing Tools (execute tests automatically)

Functional & UI Automation:

Selenium – Web automation

Cypress – Modern web test automation

Playwright – Cross-browser testing

TestComplete – GUI testing for desktop, web, and mobile

Ranorex – Cross-platform testing

Katalon Studio – Web, API, and mobile testing

Watir – Web application testing