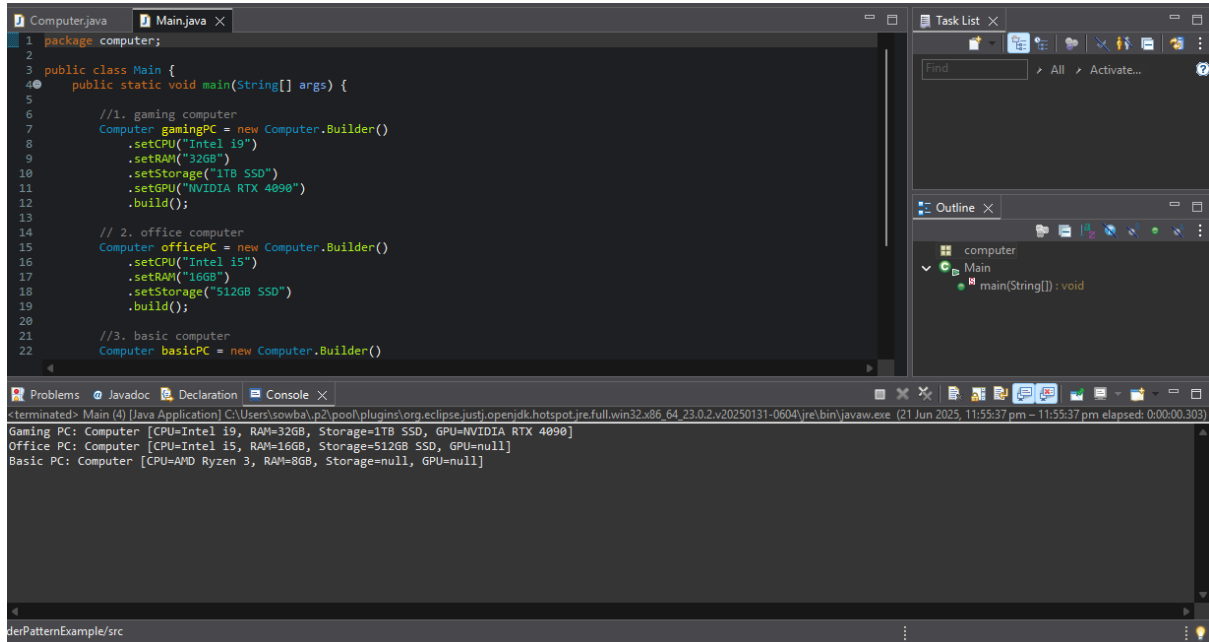


Exercise 3: Implementing the Builder Pattern

OUTPUT:



The screenshot displays an IDE with the following components:

- Editor:** Shows `Computer.java` and `Main.java`. The code defines a `Computer` class with a `Builder` inner class. The `Builder` class has methods to set `cpu`, `ram`, `storage`, and `gpu`, and a `build()` method that returns a `Computer` object. The `Main` class has a `main` method that creates three `Computer` objects: `gamingPC`, `officePC`, and `basicPC`, each using the `Builder` pattern.
- Task List:** Shows a list of tasks, including `Find`, `All`, and `Activate...`.
- Outline:** Shows the project structure, including the `computer` package and the `Main` class.
- Problems:** Shows a list of problems, including `terminated> Main (4) [Java Application]`.
- Console:** Shows the output of the program, which prints the configuration details for the three `Computer` objects: `Gaming PC: Computer [CPU=Intel i9, RAM=32GB, Storage=1TB SSD, GPU=NVIDIA RTX 4090]`, `Office PC: Computer [CPU=Intel i5, RAM=16GB, Storage=512GB SSD, GPU=null]`, and `Basic PC: Computer [CPU=AMD Ryzen 3, RAM=8GB, Storage=null, GPU=null]`.

The Builder Pattern was used to create complex `Computer` objects with optional parts. It simplifies object construction, improves code readability, and allows flexible configuration without multiple constructors.