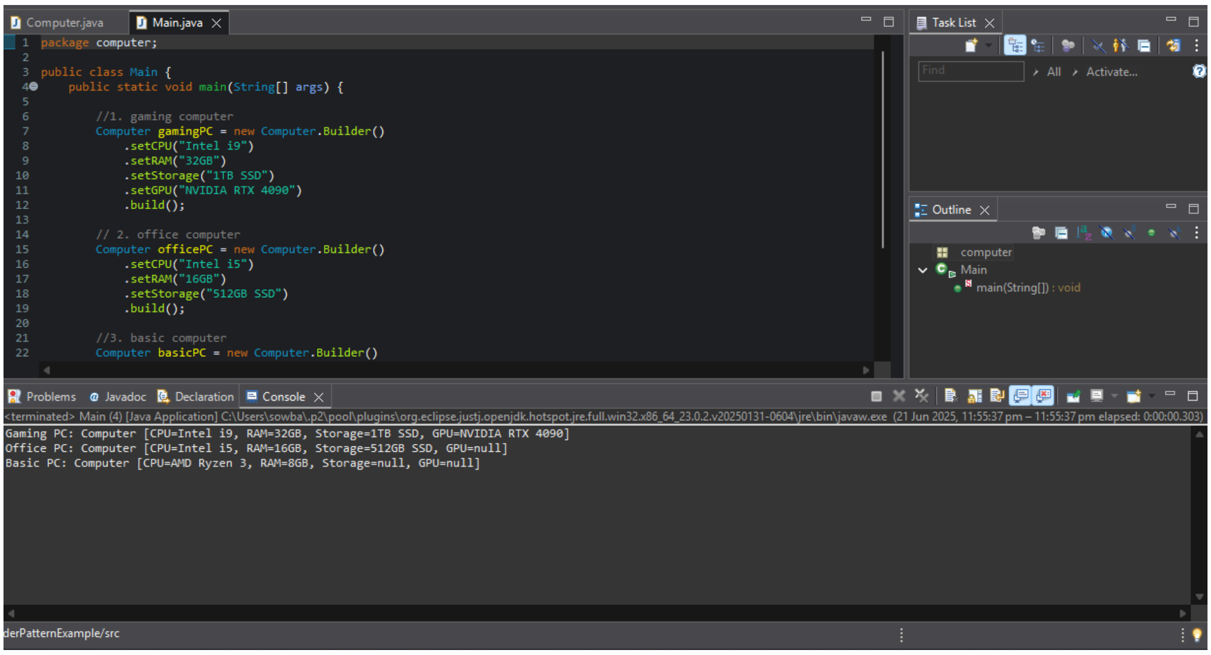
## Exercise 3: Implementing the Builder Pattern

## OUTPUT:



```java
package computer;

public class Main {
    public static void main(String[] args) {

        //1. gaming computer
        Computer gamingPC = new Computer.Builder()
            .setCPU("Intel i9")
            .setRAM("32GB")
            .setStorage("1TB SSD")
            .setGPU("NVIDIA RTX 4090")
            .build();

        // 2. office computer
        Computer officePC = new Computer.Builder()
            .setCPU("Intel i5")
            .setRAM("16GB")
            .setStorage("512GB SSD")
            .build();

        //3. basic computer
        Computer basicPC = new Computer.Builder()
```

```
<terminated> Main (4) [Java Application] C:\Users\sowba\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_23.0.2.v20250131-0604\jre\bin\javaw.exe  (21 Jun 2025, 11:55:37 pm – 11:55:37 pm elapsed: 0:00:00.303)
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]
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.