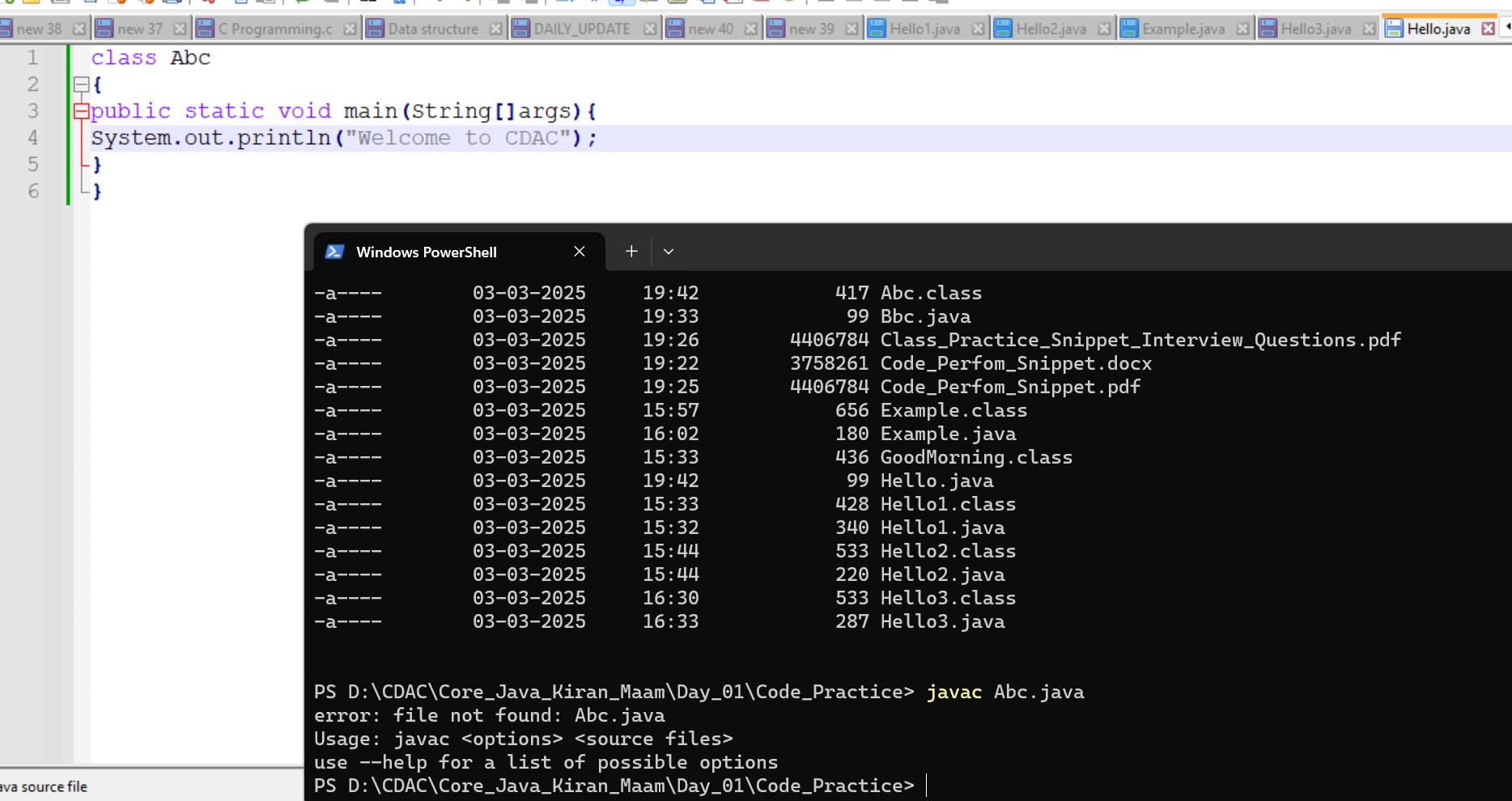
**Interview Questions & Class Notes**

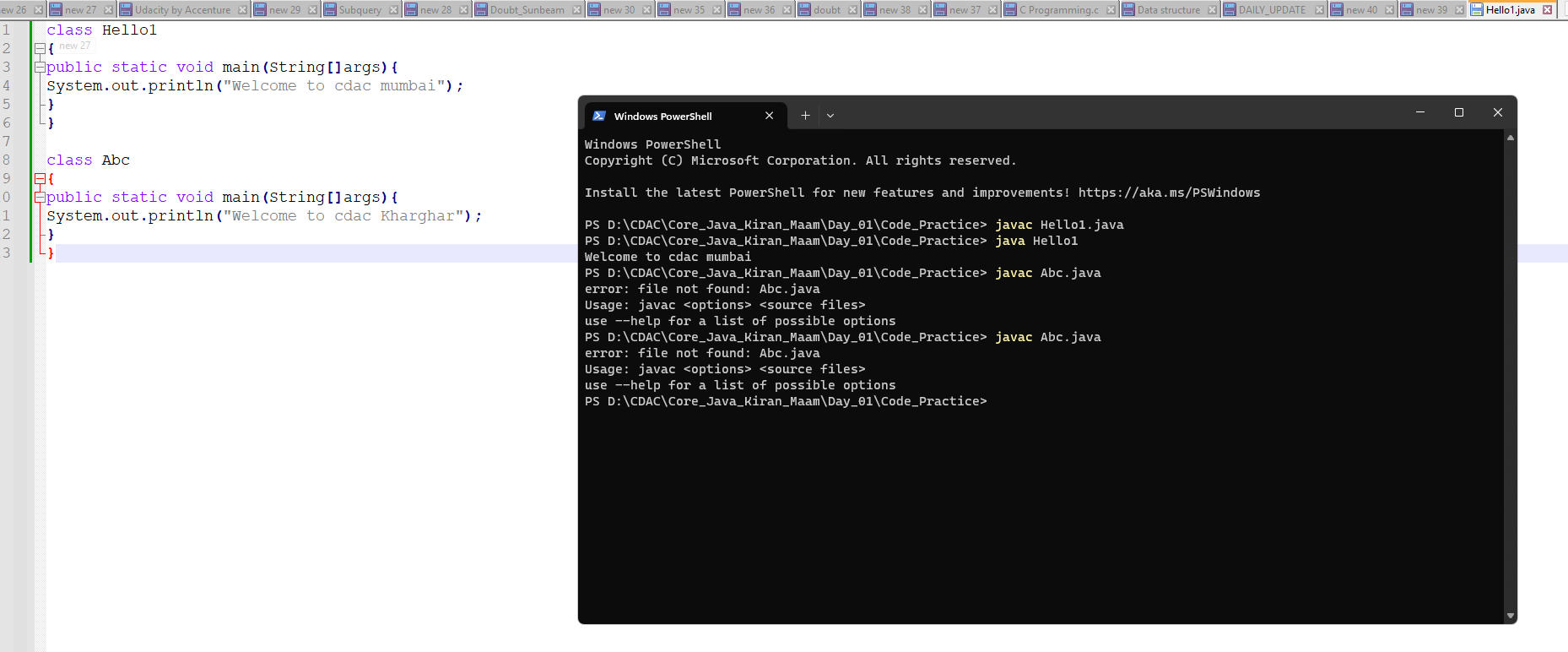
**A computer screen with white text

AI-generated content may be incorrect.**

****

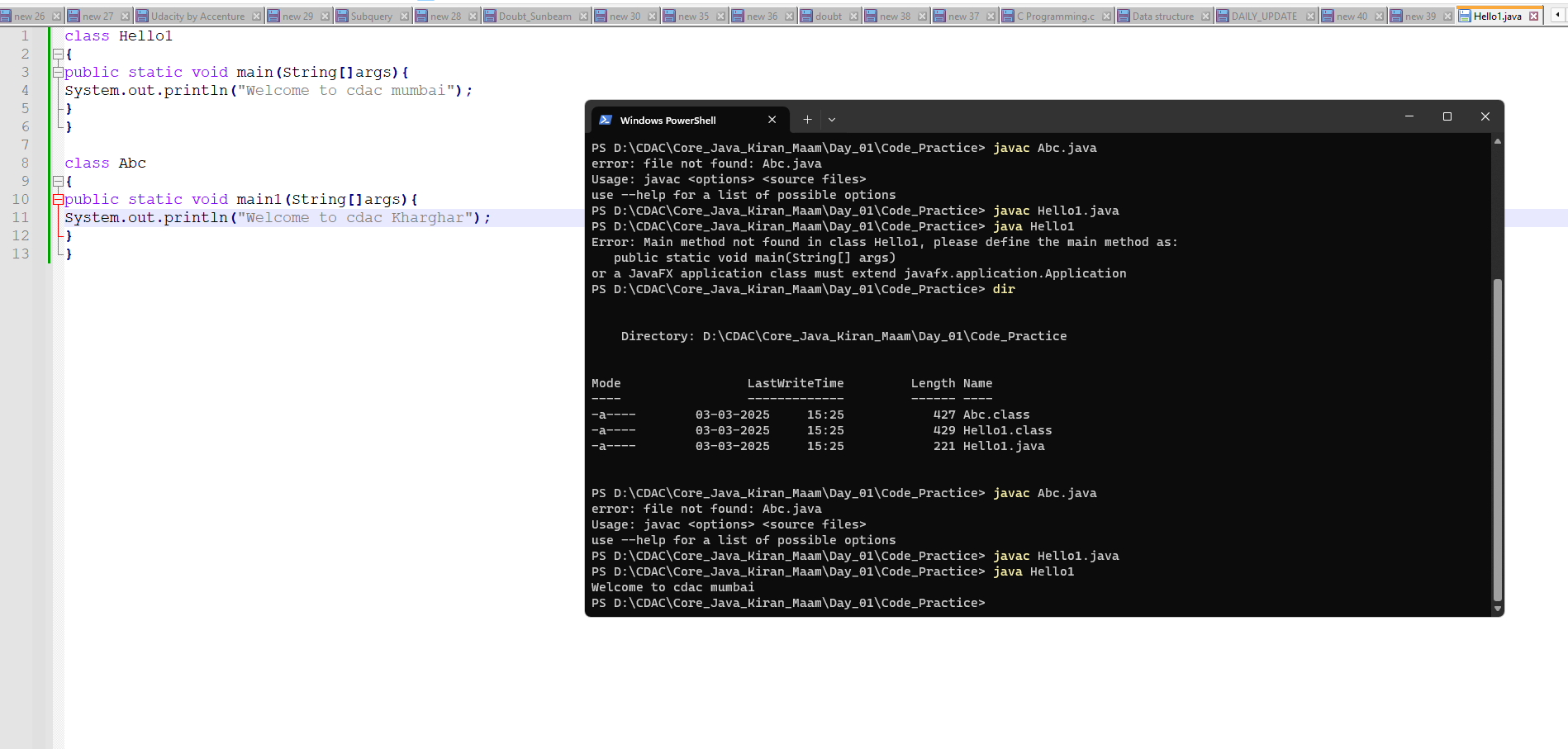
**A screenshot of a computer screen

AI-generated content may be incorrect.**



A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

Classes kitne bhi (usji sirf class file generate hogi)ho but jis nam se file save krenge vha ka hi statement execute krega.

A computer screen with a black screen

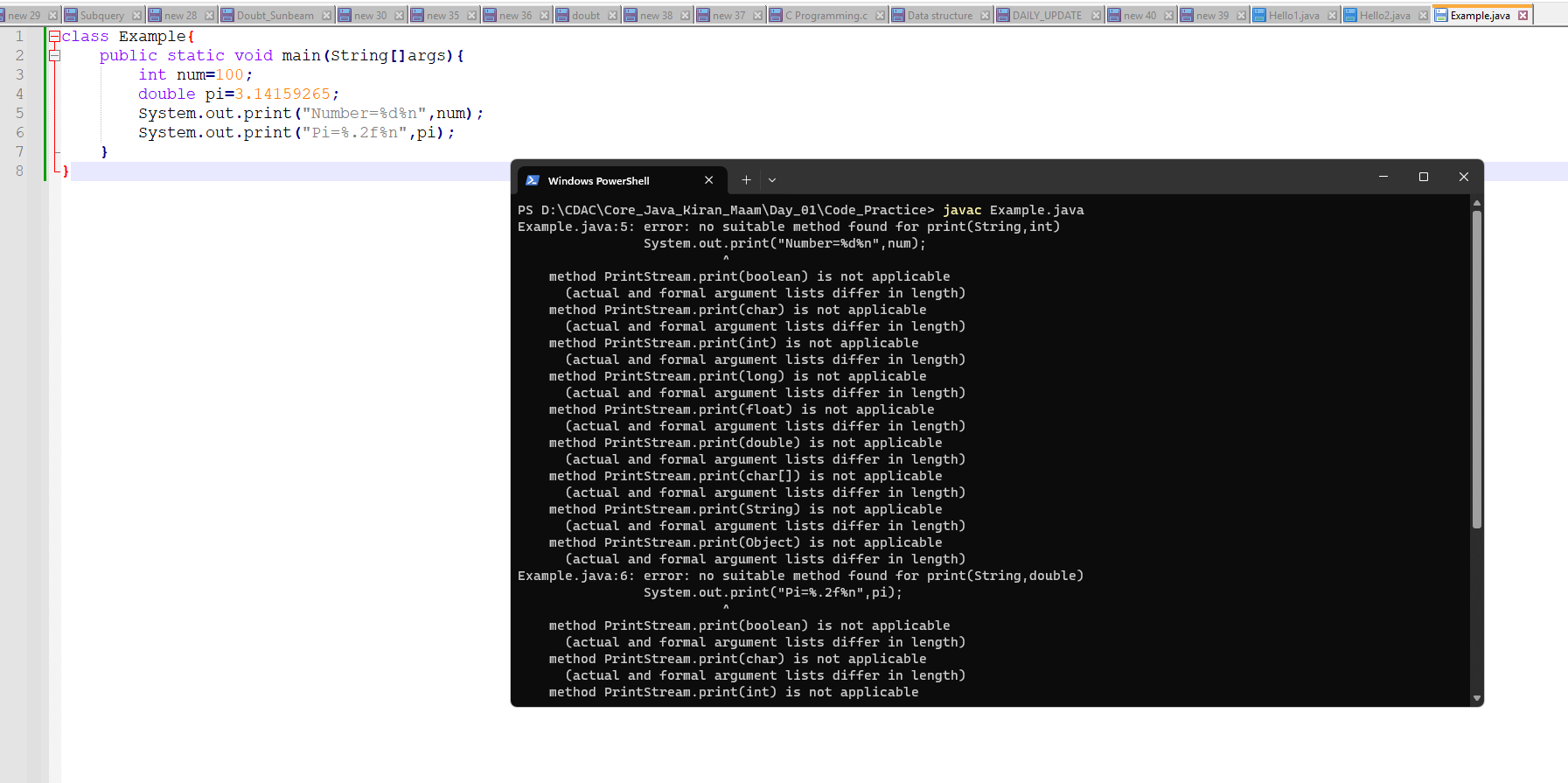
AI-generated content may be incorrect.

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.



It gives error because print it is not for formatting , print use with string blank space…

**Exercise:**

**Identify Valid and Invalid main methods from the following cases: valid : yellow , invalid : red**

public static void main(String[] args)

public static void main(String  []args)  : No matter of space

public static void main(String args[])

A screenshot of a computer program

AI-generated content may be incorrect.

public static void main(String... args)

A screen shot of a computer

AI-generated content may be incorrect.

static public void main(String[] args)

A screenshot of a computer screen

AI-generated content may be incorrect.

public static final void main(String[] args)

A screen shot of a computer

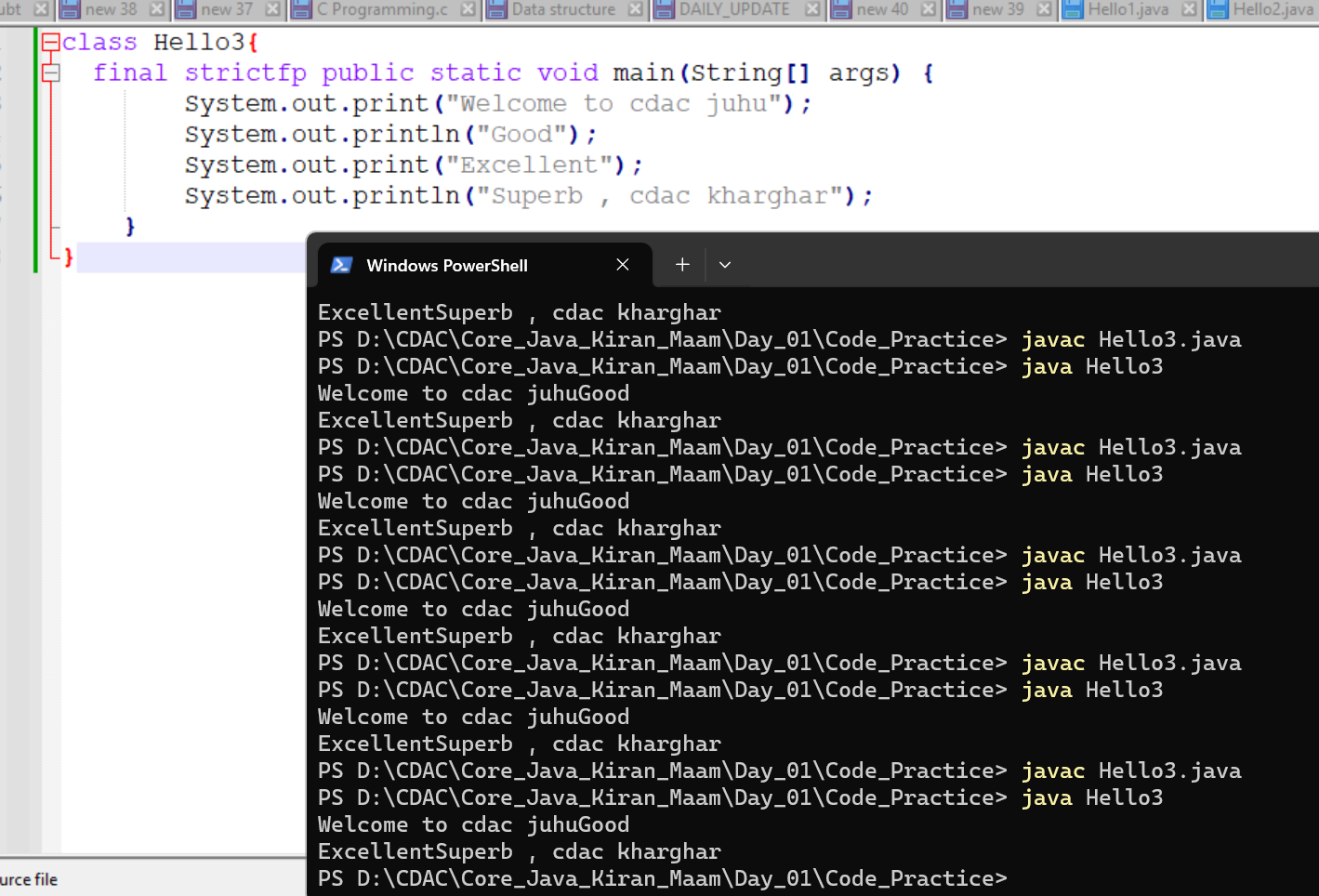
AI-generated content may be incorrect.

final public static void main(String[] args)

A screen shot of a computer screen

AI-generated content may be incorrect.

final strictfp public static void main(String[] args)



strictfp is unnecessary unless you are working with floating-point operations that need strict adherence to IEEE 754.

public void main(String[] args)   : Runtime error

A screen shot of a computer screen

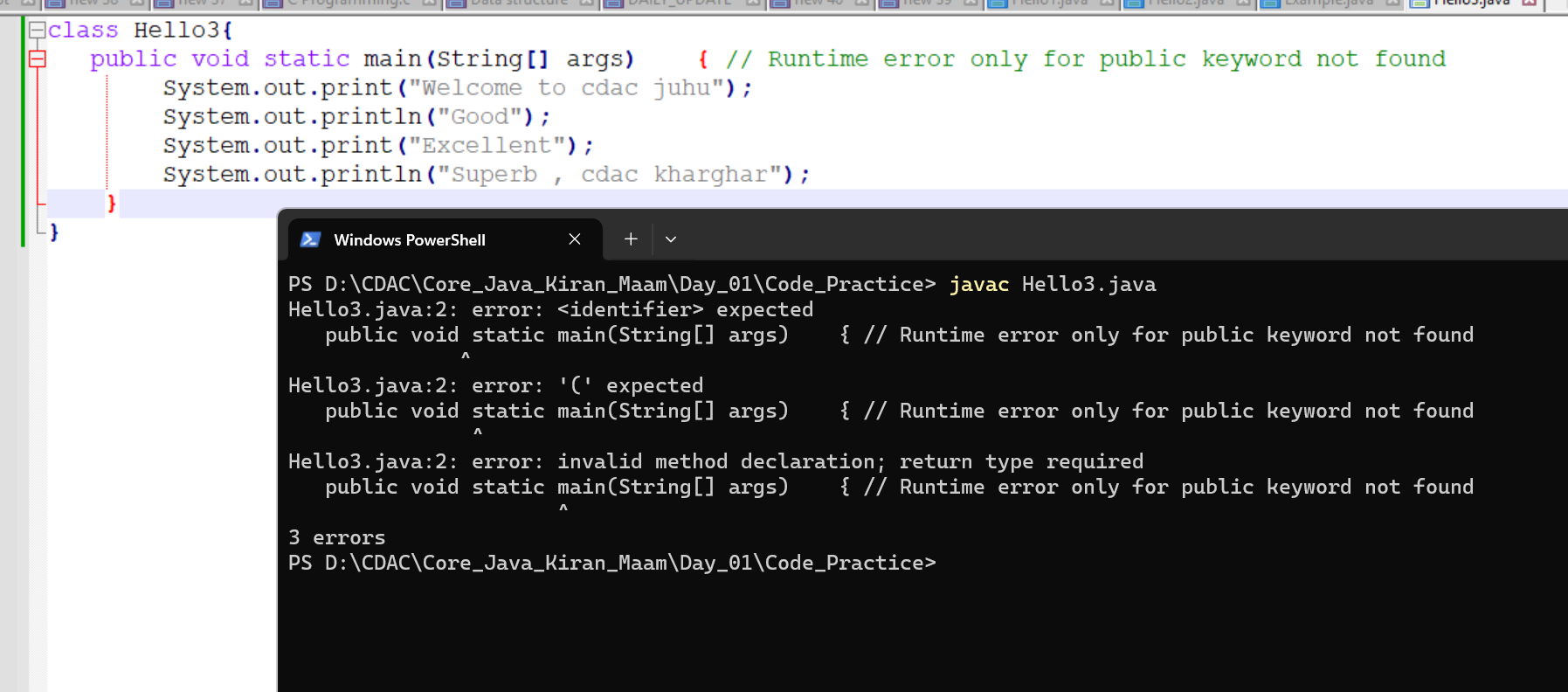
AI-generated content may be incorrect.

static void main(String[] args)

A screen shot of a computer screen

AI-generated content may be incorrect.

public void static main(String[] args)



Not runtime error compile time error

abstract public static void main(String[] args)  : Compile time error

A screenshot of a computer screen

AI-generated content may be incorrect.

**Interview Questions:**

**1. What is Java?**

Java is a high-level, object-oriented programming language developed by **Sun Microsystems** (now owned by Oracle). It is designed to be **platform-independent**, **secure**, **robust**, and **multithreaded**, making it ideal for building various applications such as web, mobile, and enterprise applications.

**2. Explain the features of Java.**

Java offers several powerful features:

* **Platform Independence** – Java programs can run on any system with a JVM (**Write Once, Run Anywhere - WORA**).
* **Object-Oriented** – Everything in Java is based on objects and classes.
* **Robust** – Strong memory management, exception handling, and garbage collection.
* **Multi-threaded** – Supports concurrent execution of multiple tasks.
* **Secure** – Uses a security manager to define access rules for classes.
* **Automatic Garbage Collection** – JVM automatically manages memory using garbage collection.
* **Interpreted and Compiled** – Java is compiled into bytecode, which is then interpreted by the JVM.
* **High Performance** – Uses Just-In-Time (JIT) compiler to optimize execution.
* **Dynamic and Extensible** – Supports dynamic memory allocation and external libraries.

**3. What is the difference between JDK, JRE, and JVM?**

| **Component** | **Description** |
| --- | --- |
| **JDK (Java Development Kit)** | A software development kit that includes JRE + development tools like compiler (javac), debugger, and JavaDoc. |
| **JRE (Java Runtime Environment)** | Provides the necessary libraries and JVM to run Java applications but does not include development tools. |
| **JVM (Java Virtual Machine)** | An abstract machine that runs Java bytecode by converting it into machine code for execution. |

👉 **JDK = JRE + Development Tools**  
👉 **JRE = JVM + Java Libraries**

**4. Why is Java platform-independent?**

Java is platform-independent because of the **JVM (Java Virtual Machine)**. Java source code is compiled into **bytecode** (not machine-specific). This bytecode can run on any system with a compatible JVM, making Java applications portable.

**5. What is the difference between compiled and interpreted languages?**

| **Feature** | **Compiled Language** | **Interpreted Language** |
| --- | --- | --- |
| **Execution** | Converts entire code to machine code before execution. | Translates and executes line by line. |
| **Speed** | Faster since it runs machine code directly. | Slower due to real-time translation. |
| **Example Languages** | C, C++ | Python, JavaScript |
| **Error Detection** | Errors are detected at compile-time. | Errors appear at runtime. |

**Java is both compiled and interpreted:**

* The **javac** compiler converts Java code into **bytecode**.
* The **JVM** interprets bytecode into machine code during execution.

**6. What is the difference between C++ and Java?**

| **Feature** | **C++** | **Java** |
| --- | --- | --- |
| **Platform** | Platform-dependent | Platform-independent (JVM) |
| **Memory Management** | Manual (pointers) | Automatic (Garbage Collection) |
| **Multiple Inheritance** | Supported | Not supported (uses interfaces) |
| **Object-Oriented** | Partially (supports procedural programming) | Fully object-oriented |
| **Compilation & Execution** | Directly compiled to machine code | Compiled to bytecode and executed via JVM |
| **Exception Handling** | No built-in exception handling | Built-in exception handling |

**7. What are the four pillars of OOP in Java?**

1. **Encapsulation**
   * Wrapping data and methods into a single unit (class).
   * Example: Using **private variables** and providing access via **getter/setter** methods.
2. **Abstraction**
   * Hiding implementation details and exposing only essential features.
   * Example: Using **abstract classes** and **interfaces**.
3. **Inheritance**
   * Enabling one class to inherit properties and behaviors from another.
   * Example: class Child extends Parent {}
4. **Polymorphism**
   * Allowing the same method to have multiple implementations.
   * **Compile-time polymorphism (Method Overloading)**
   * **Runtime polymorphism (Method Overriding)**

**8. What is Object-Oriented Programming (OOP)?**

OOP (**Object-Oriented Programming**) is a **programming paradigm** that models real-world entities using **objects** and **classes**. It enables **modularity, reusability, scalability, and easier maintenance** of code.

**Key Concepts of OOP in Java:**

1. **Objects** – Instances of classes that represent real-world entities.
2. **Classes** – Blueprints or templates for creating objects.
3. **Methods** – Define behaviors/actions an object can perform.
4. **Attributes (Fields/Properties)** – Define the state of an object.

**Benefits of OOP:**

**Code Reusability** – Inheritance allows code reuse.  
 **Modularity** – Code is divided into objects, making it manageable.  
 **Flexibility & Scalability** – Easily extend features using polymorphism.  
 **Security** – Encapsulation hides data from unauthorized access.