ABSTRACTION

BY USING **INTERFACE**

IN THE NAME OF ALLAH

بسم الله الرحمن الرحيم

THE GRACIOUS, THE MERCIFUL.

GETS - GETTING EDUCATION WITH TECHNOLOGICAL SYSTEM

Instructor: Sir A.Rehman Ali Brohi

LECTURE: 10
Abstraction in java (by using Interface)

How to achieve abstraction in java

1. By Using Abstract Class. (0 % 100)
2. By Using Interfaces. (%100)

In this class we will learn how to achieve Abstraction using Interface

What is Concrete Method?

Concrete methods in Java are nothing but just like any other normal methods. The **methods which are not abstract methods** are called to be concrete methods in java.

Interfaces are similar to Abstract Class but having all the methods of abstract type.

abstract class Test {

1. Abstract void show();
2. Void display(){

   }

}

Interface interface1 {

//abstract method

not concrete method means not with body method

Not about 8 or 9th version of java

}

USE OF INTERFACE

1. It is used to achieve abstraction.
2. It support multiple inheritance.
3. It can be used to achieve loose coupling.

SYNTEX OF INTERFACE

Interface InterfaceName{

1. **Methods**

//compiler auto add **abstract public**

2. **Fields**

//compiler auto add **public, static final**

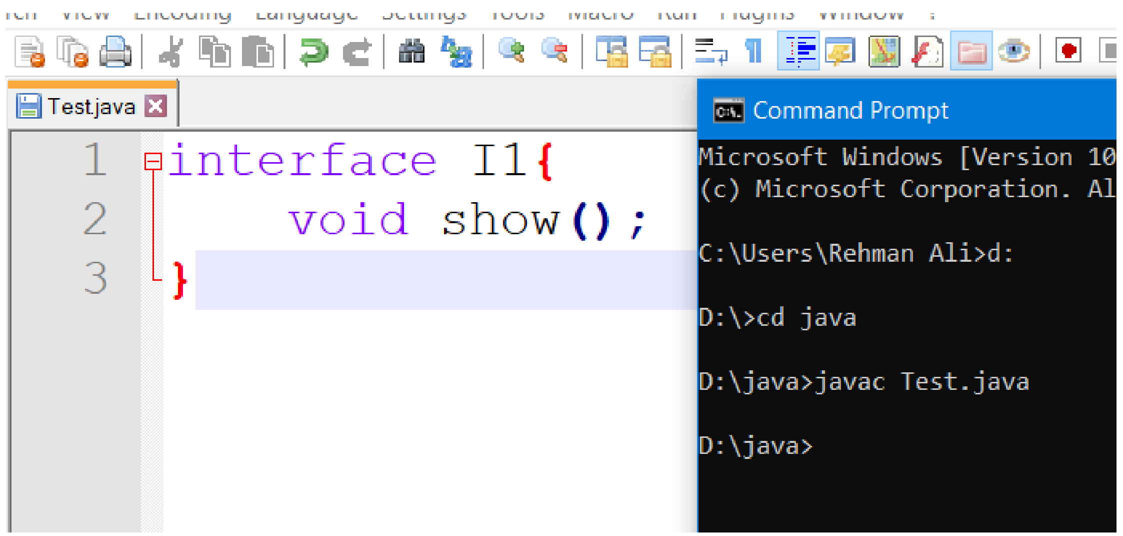
}

3. **8th version**:
   - Default concrete method
   - Static methods

4. **9th version:**
   - Private methods

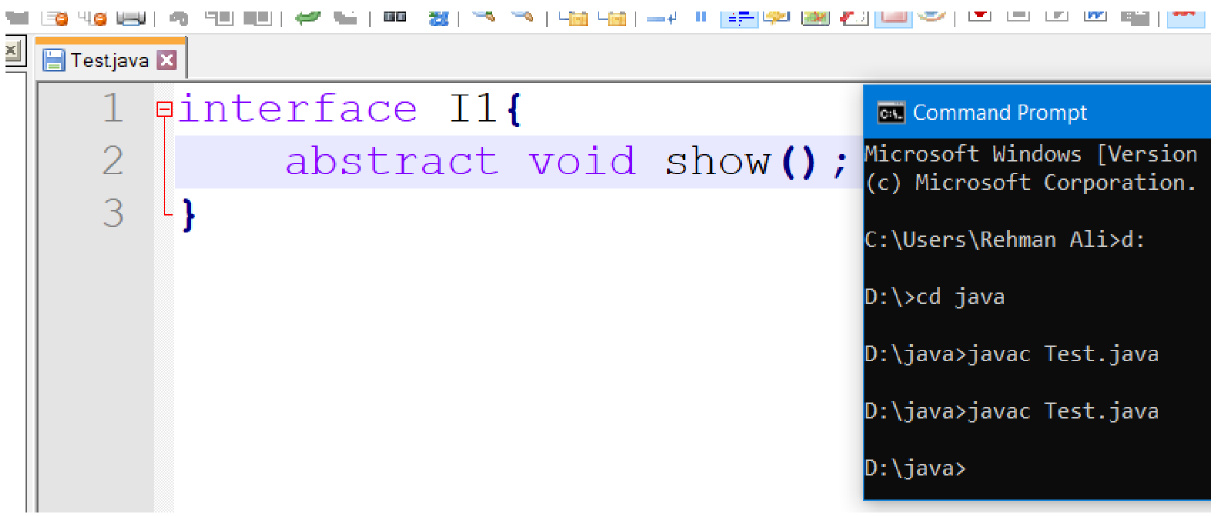
```
Interface InterfaceProgram{

    public abstract       void show();

    public static final     int a = 10;

    Default void display(){

    }

    static void run(){

    }
}
```

Understanding about compiling and errors



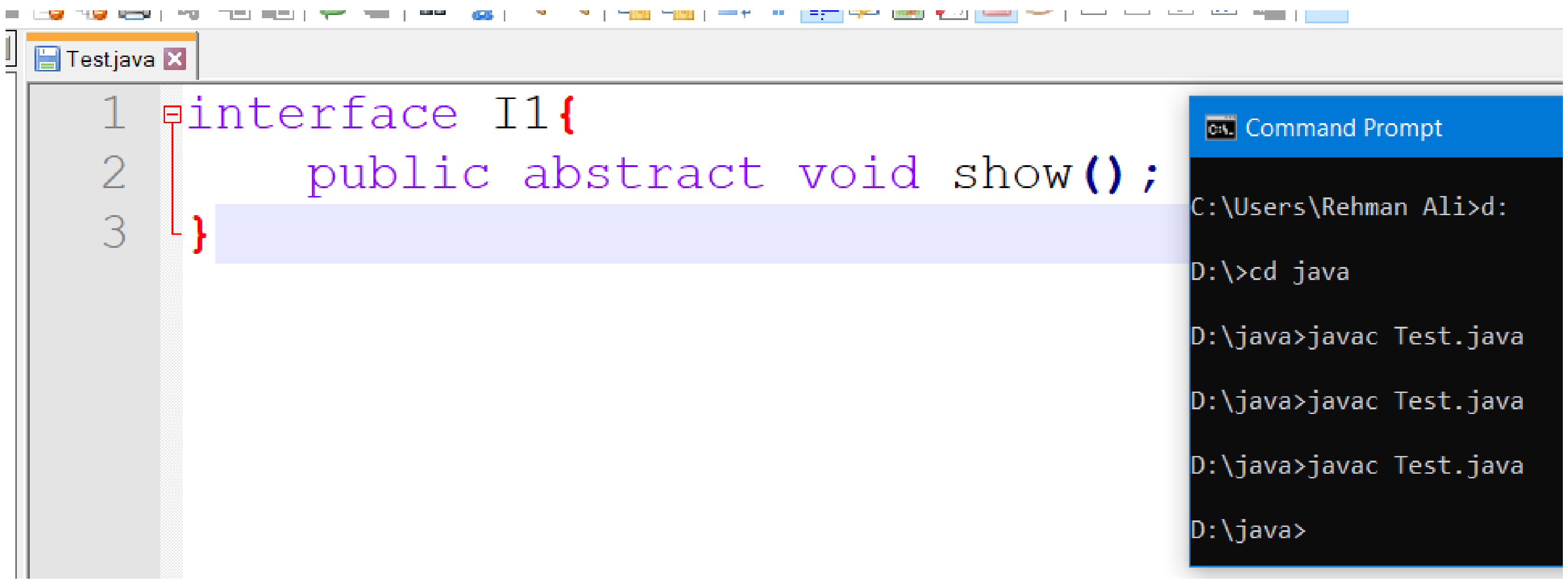
```java
interface I1{
        void show();
}
```

```
Microsoft Windows [Version 10
(c) Microsoft Corporation. Al

C:\Users\Rehman Ali>d:

D:\>cd java

D:\java>javac Test.java

D:\java>
```



```java
interface I1{
        abstract void show();
}
```

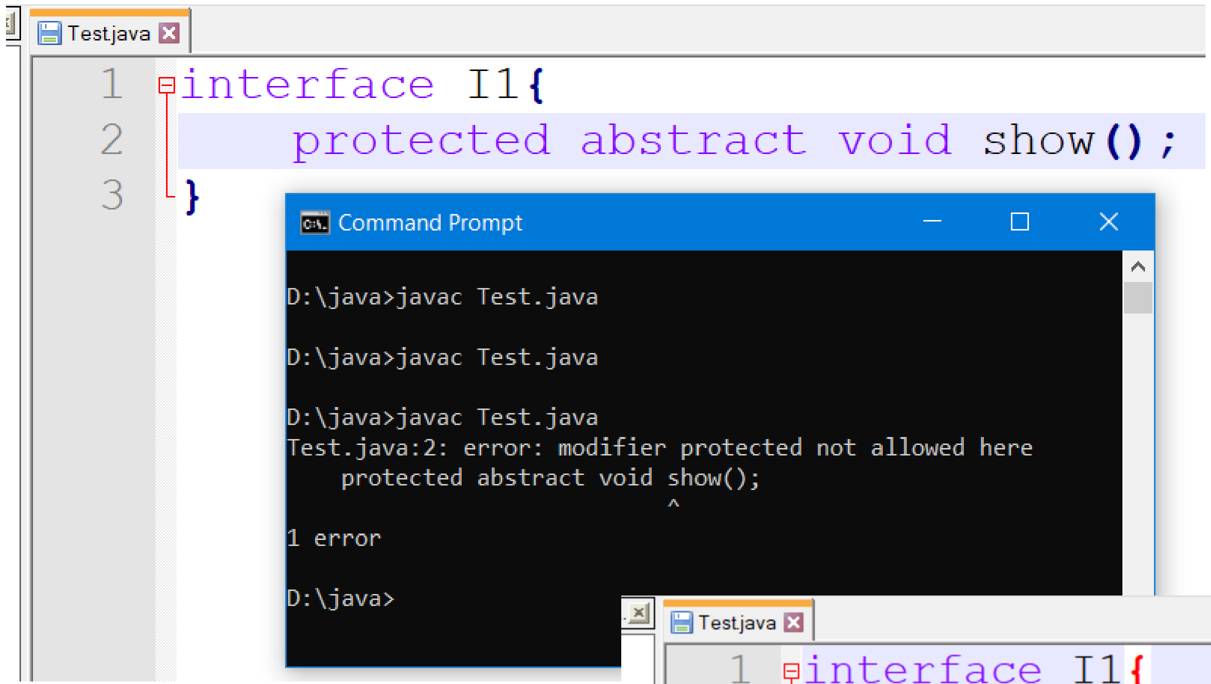
```
Microsoft Windows [Version
(c) Microsoft Corporation.

C:\Users\Rehman Ali>d:

D:\>cd java

D:\java>javac Test.java

D:\java>javac Test.java

D:\java>
```



```java
interface I1{
        public abstract void show();
}
```

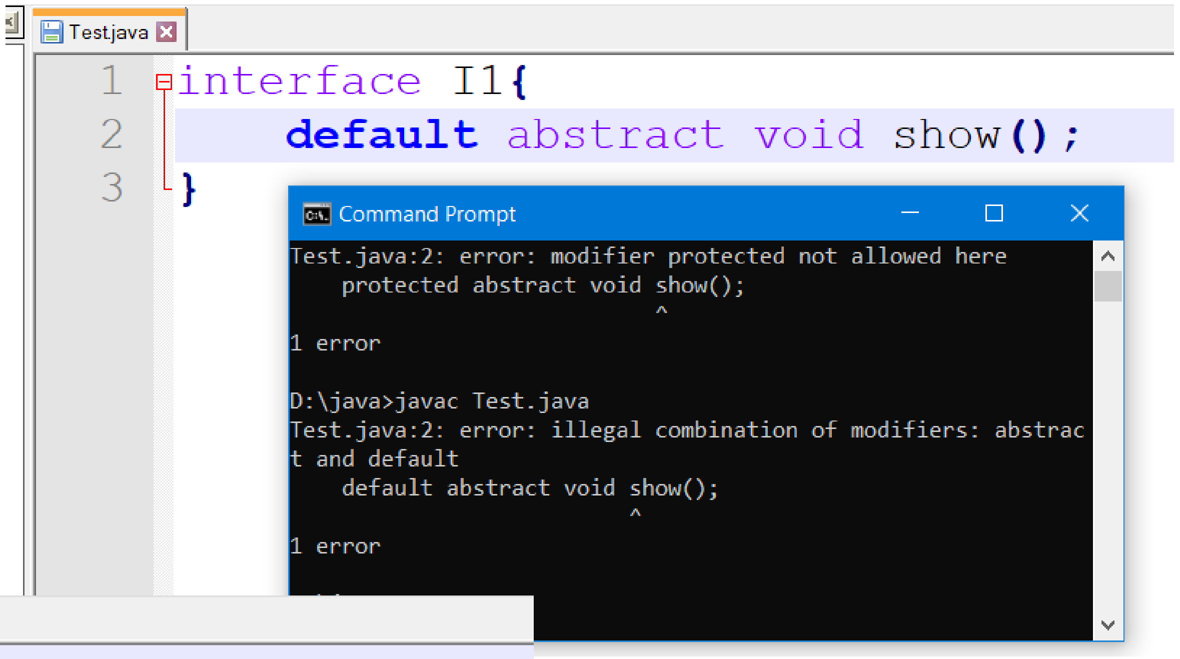
```
C:\Users\Rehman Ali>d:

D:\>cd java

D:\java>javac Test.java

D:\java>javac Test.java

D:\java>javac Test.java

D:\java>
```

Understanding about compiling and errors



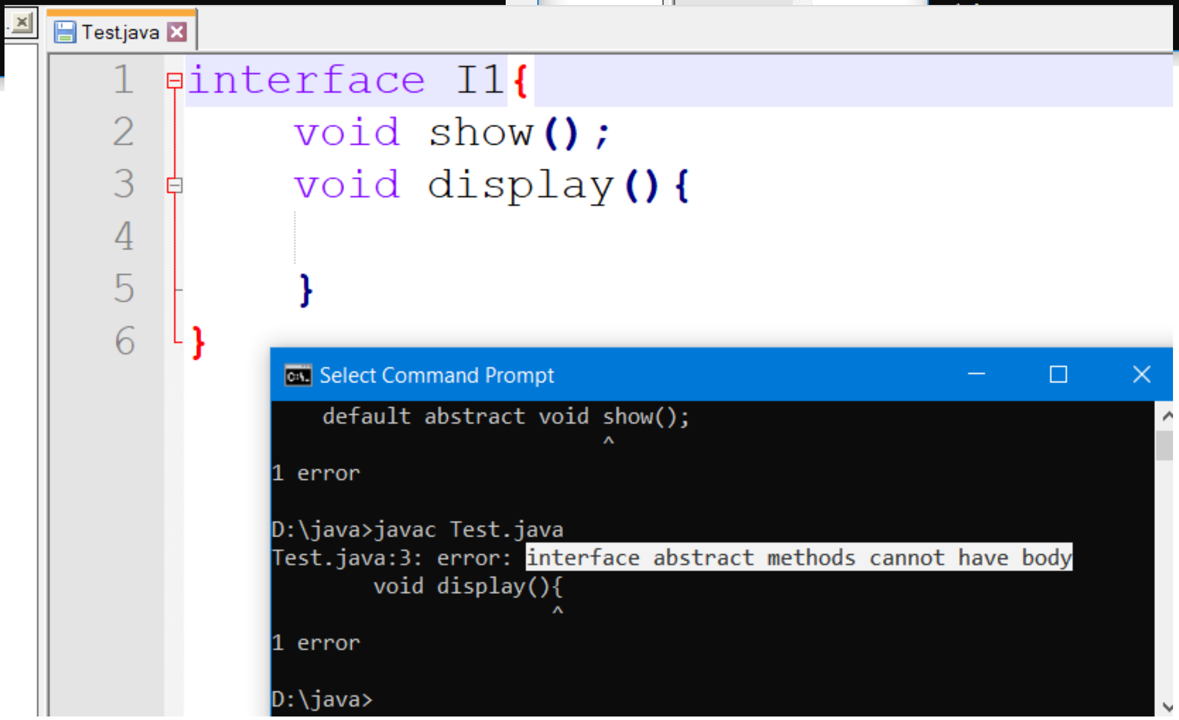
```java
interface I1{
	protected abstract void show();
}
```

```
D:\java>javac Test.java

D:\java>javac Test.java

D:\java>javac Test.java
Test.java:2: error: modifier protected not allowed here
    protected abstract void show();
                            ^
1 error

D:\java>
```



```java
interface I1{
	default abstract void show();
}
```

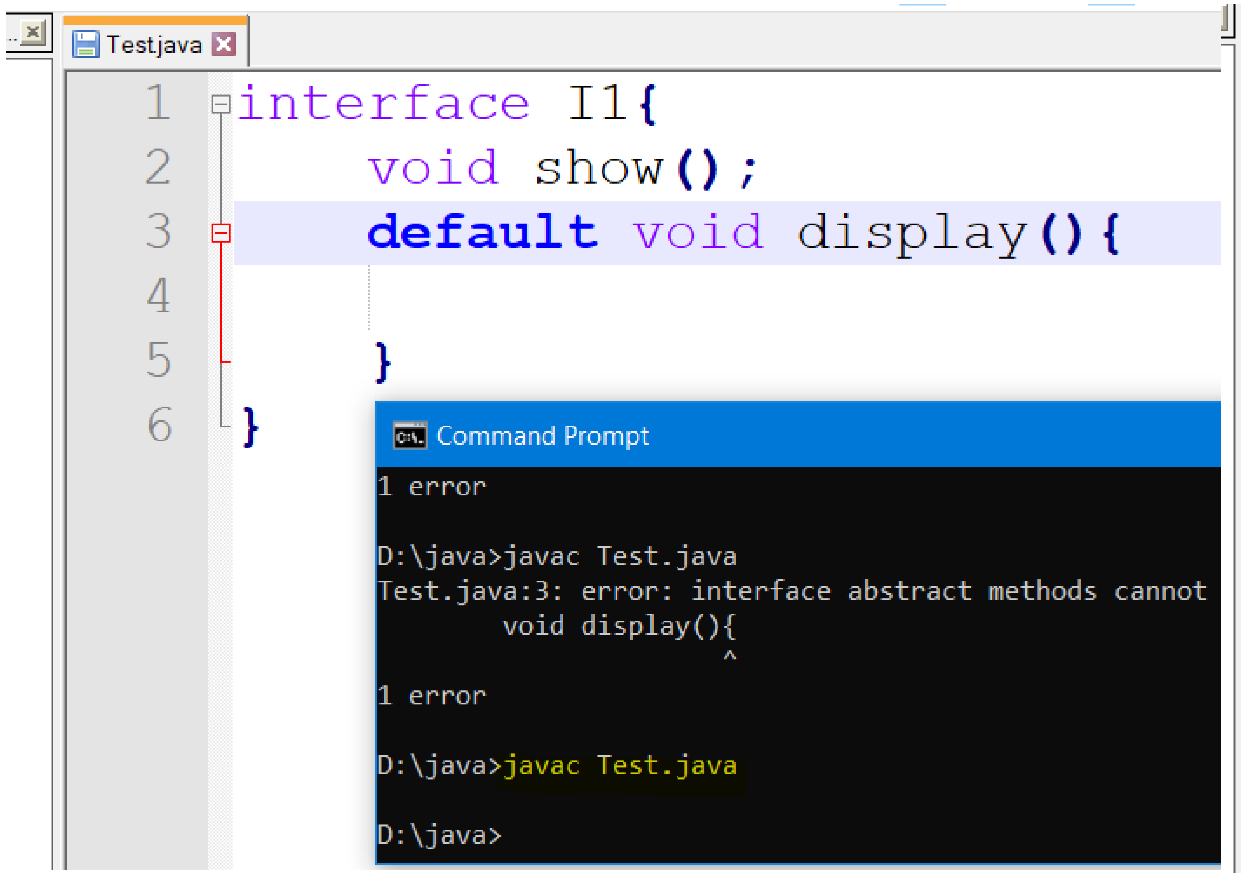
```
Test.java:2: error: modifier protected not allowed here
    protected abstract void show();
                            ^
1 error

D:\java>javac Test.java
Test.java:2: error: illegal combination of modifiers: abstrac
t and default
    default abstract void show();
                          ^
1 error
```



```java
interface I1{
	void show();
	void display(){

	}
}
```

```
    default abstract void show();
                          ^
1 error

D:\java>javac Test.java
Test.java:3: error: interface abstract methods cannot have body
        void display(){
                      ^
1 error

D:\java>
```

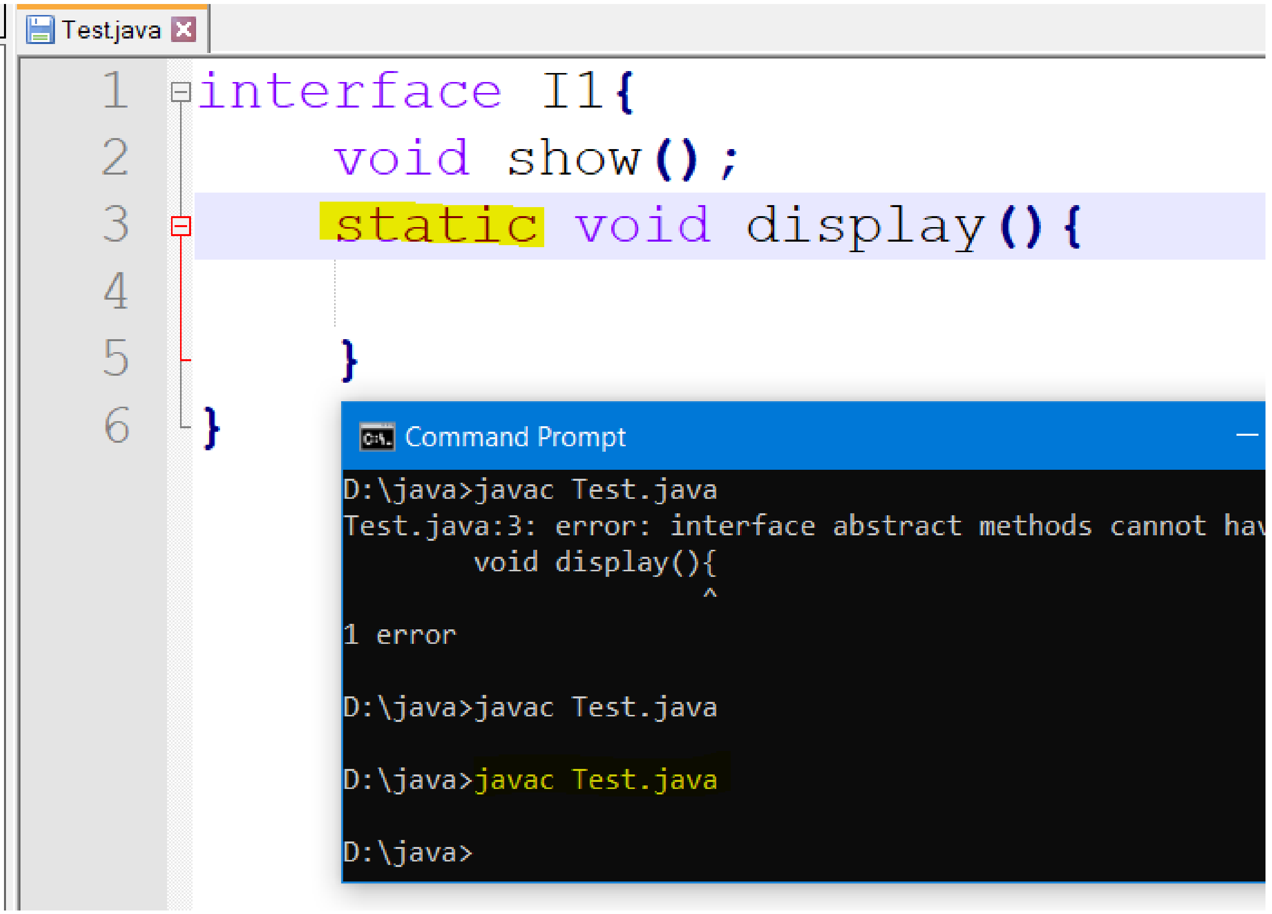
Understanding about compiling and errors



Test.java

```java
interface I1{
	void show();
	default void display(){

	}
}
```

```
1 error

D:\java>javac Test.java
Test.java:3: error: interface abstract methods cannot
        void display(){
                      ^
1 error

D:\java>javac Test.java

D:\java>
```

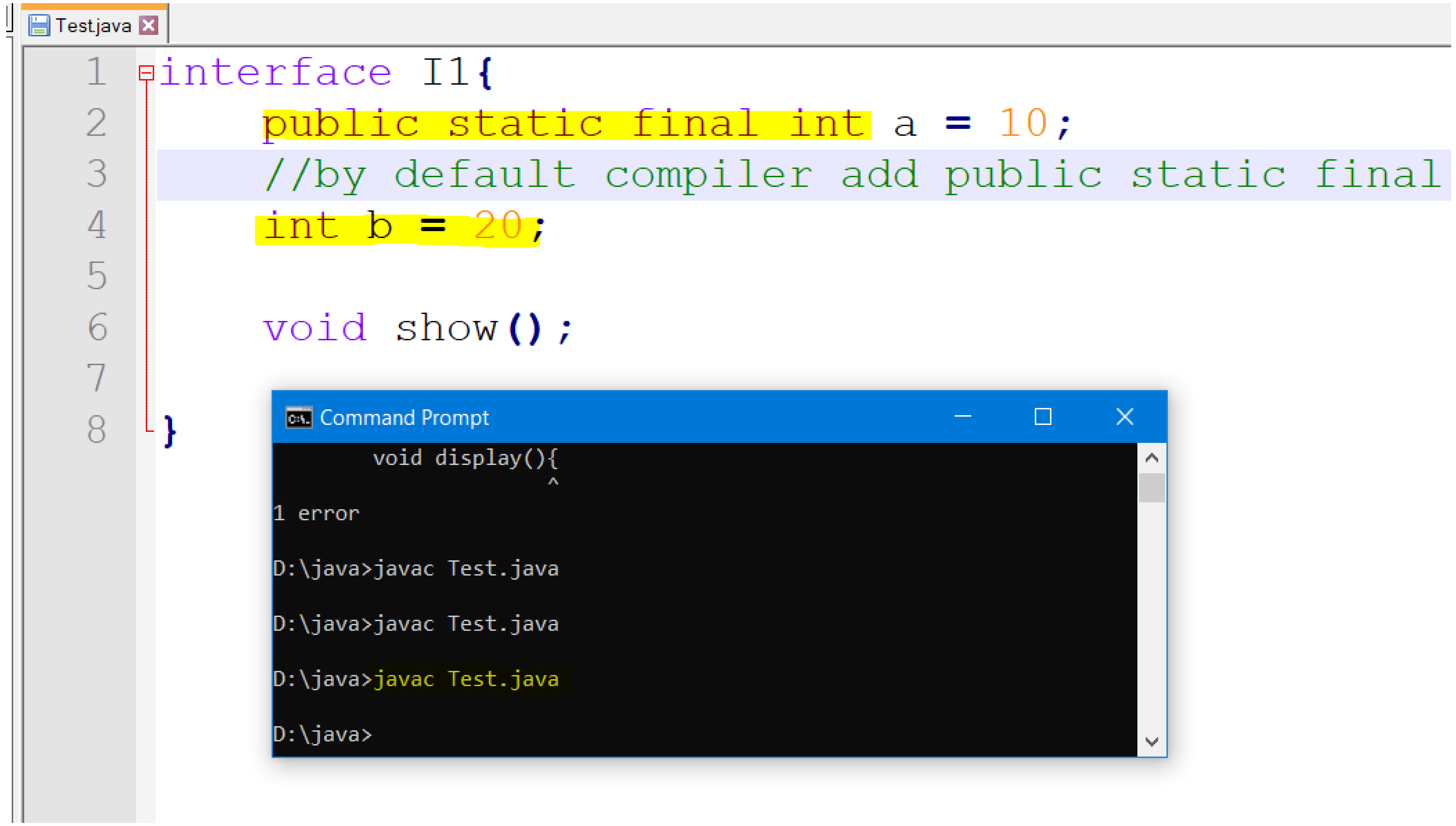


Test.java

```java
interface I1{
	void show();
	static void display(){

	}
}
```

```
D:\java>javac Test.java
Test.java:3: error: interface abstract methods cannot hav
        void display(){
                      ^
1 error

D:\java>javac Test.java

D:\java>javac Test.java

D:\java>
```

Understanding about compiling and errors



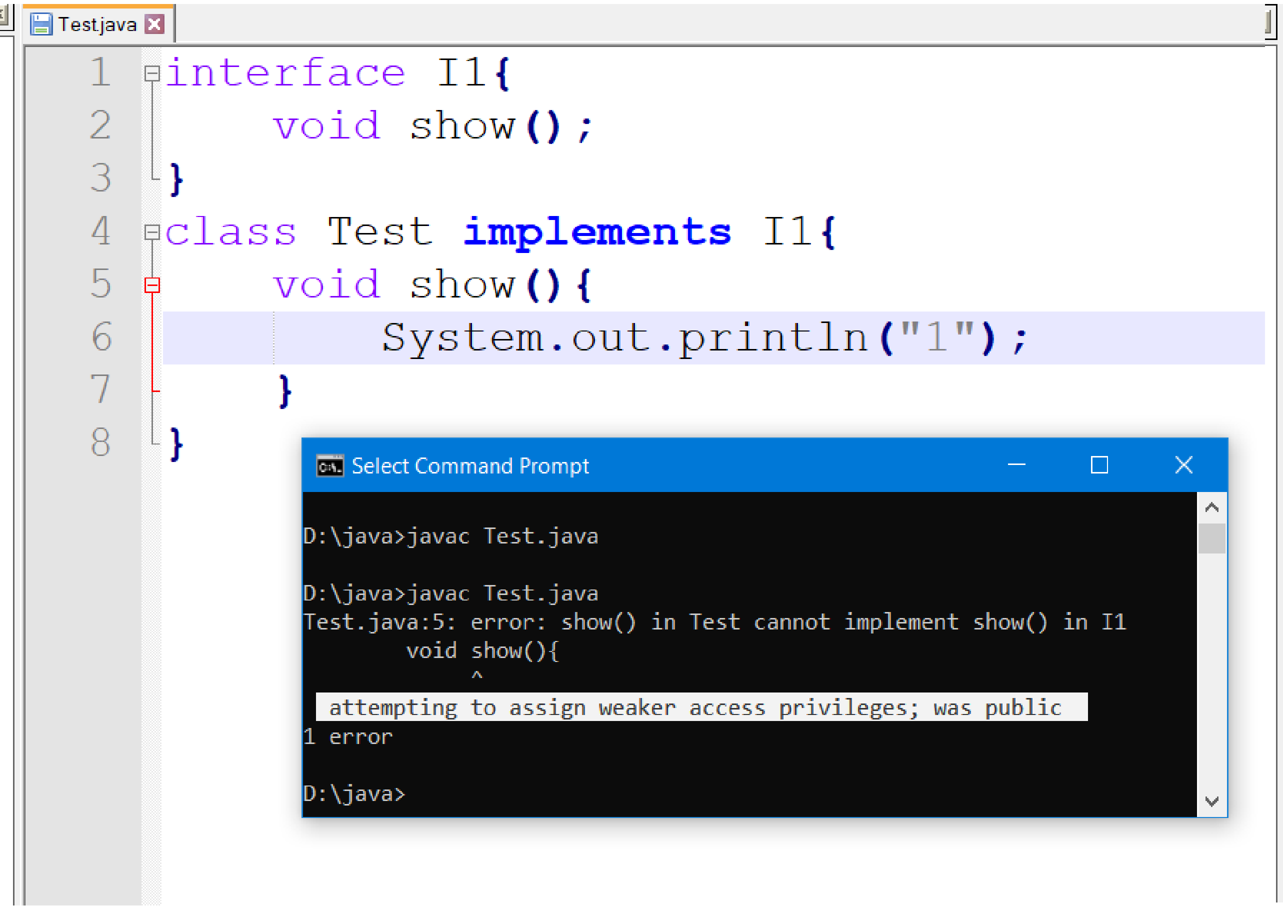
Test.java

```
interface I1{
	public static final int a = 10;
	//by default compiler add public static final
	int b = 20;

	void show();

}
```

Command Prompt

```
      void display(){
                    ^
1 error

D:\java>javac Test.java

D:\java>javac Test.java

D:\java>javac Test.java

D:\java>
```

Abstraction Program

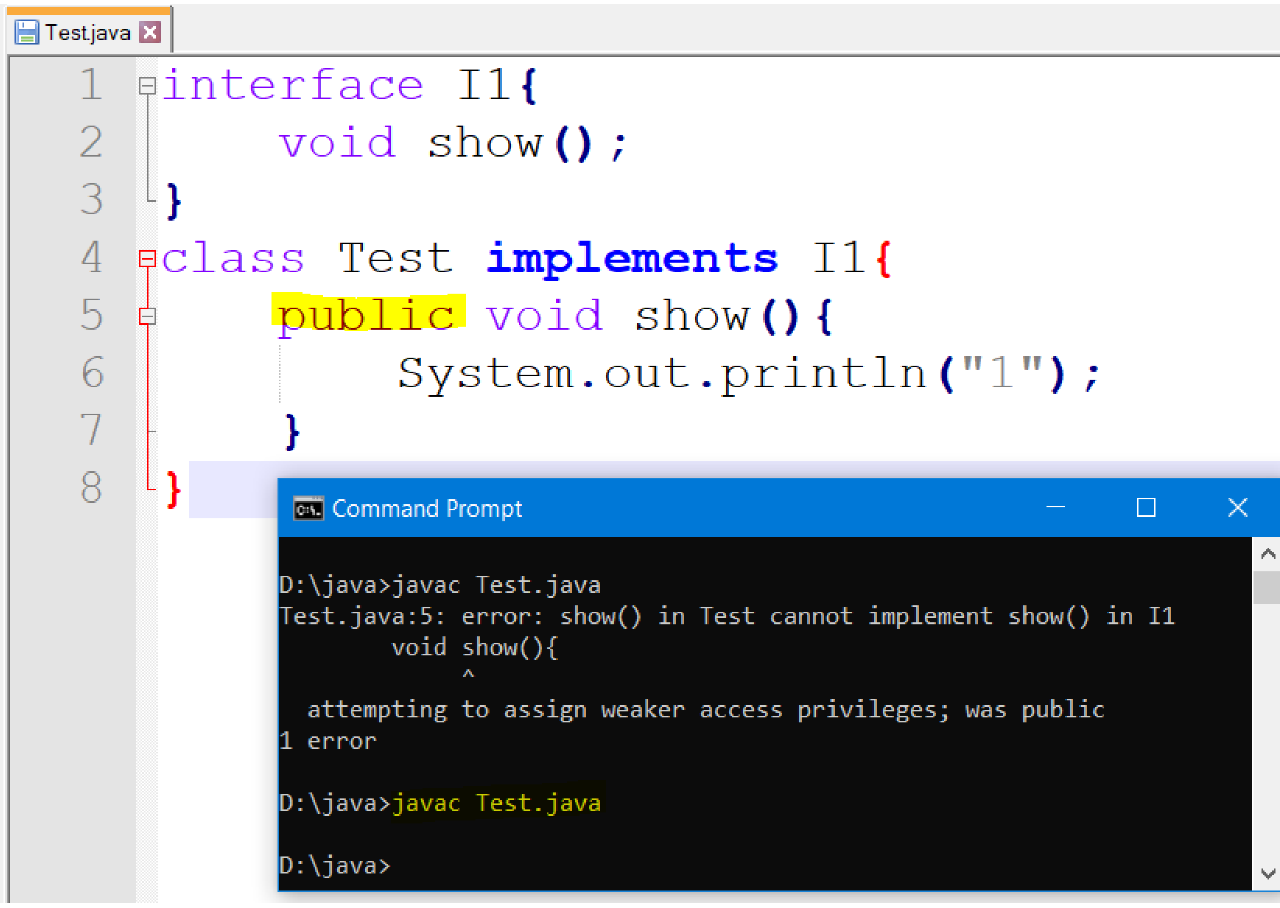


Test.java

```java
interface I1{
	void show();
}
class Test implements I1{
	void show(){
		System.out.println("1");
	}
}
```

Select Command Prompt

```
D:\java>javac Test.java

D:\java>javac Test.java
Test.java:5: error: show() in Test cannot implement show() in I1
        void show(){
             ^
  attempting to assign weaker access privileges; was public
1 error

D:\java>
```



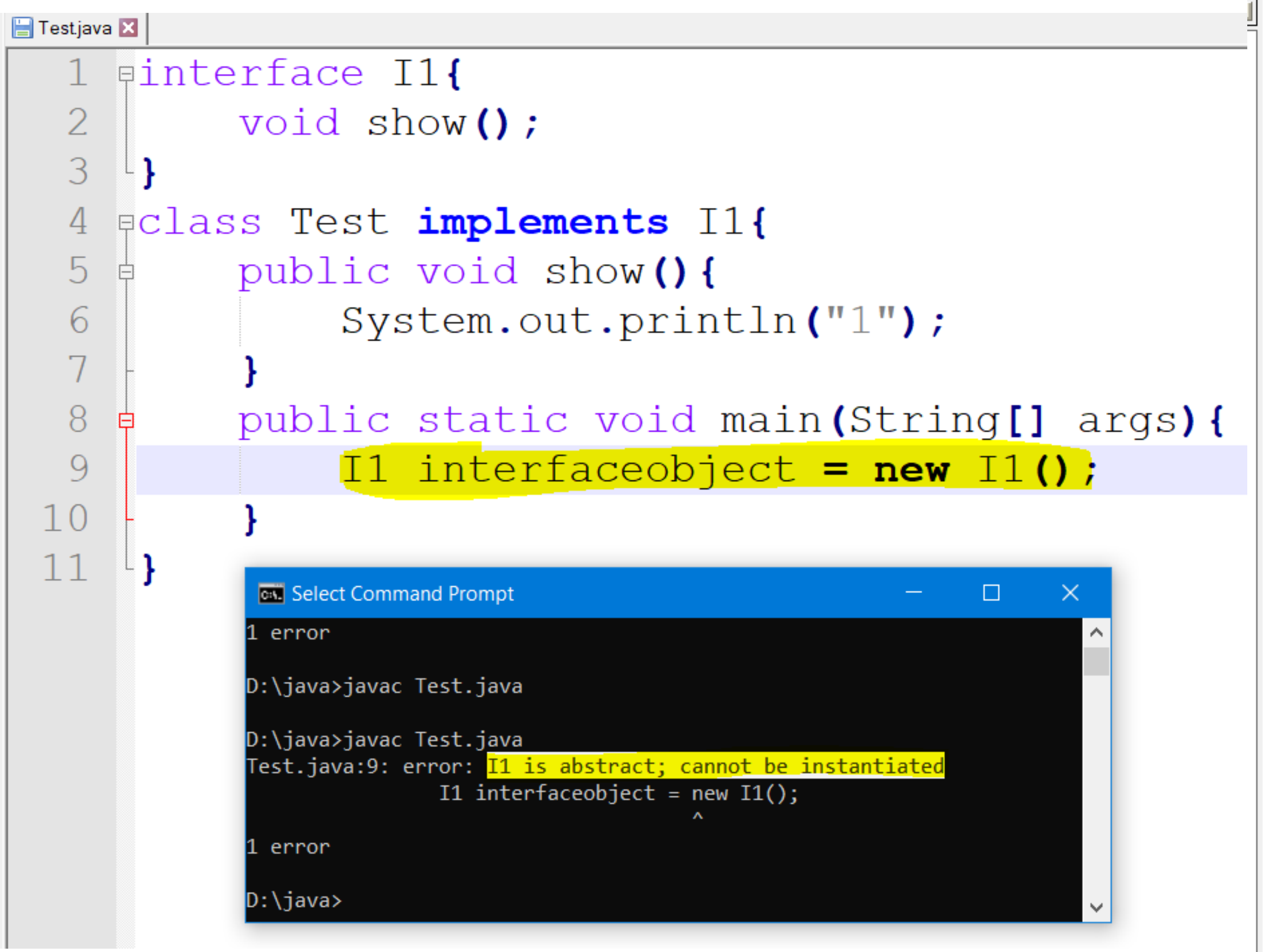
Test.java

```java
interface I1{
	void show();
}
class Test implements I1{
	public void show(){
		System.out.println("1");
	}
}
```

Command Prompt

```
D:\java>javac Test.java
Test.java:5: error: show() in Test cannot implement show() in I1
        void show(){
             ^
  attempting to assign weaker access privileges; was public
1 error

D:\java>javac Test.java

D:\java>
```

Abstraction Program

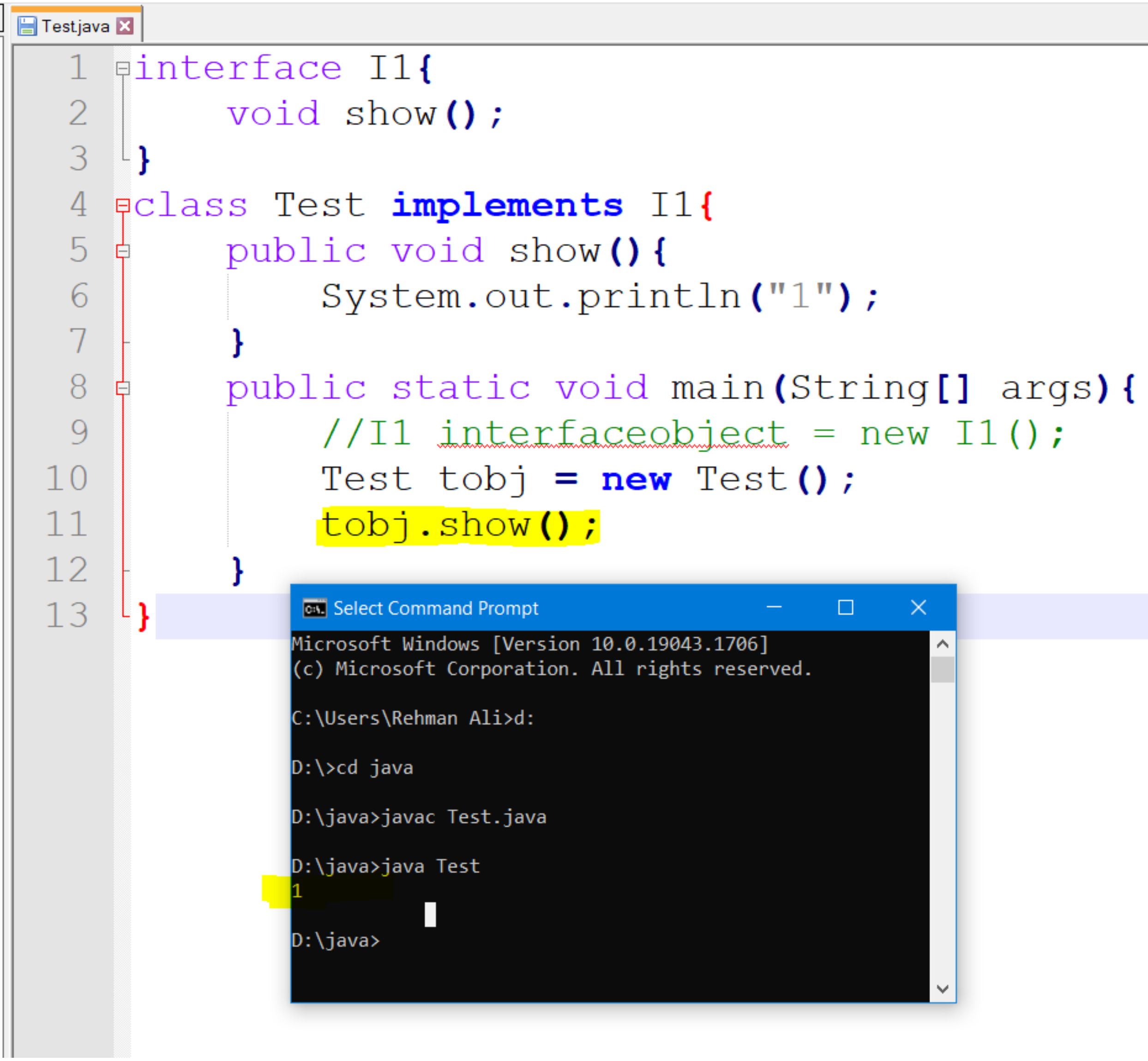


Test.java

```java
interface I1{
	void show();
}
class Test implements I1{
	public void show(){
		System.out.println("1");
	}
	public static void main(String[] args){
		I1 interfaceobject = new I1();
	}
}
```

Select Command Prompt

```
1 error

D:\java>javac Test.java

D:\java>javac Test.java
Test.java:9: error: I1 is abstract; cannot be instantiated
                I1 interfaceobject = new I1();
                                     ^
1 error

D:\java>
```



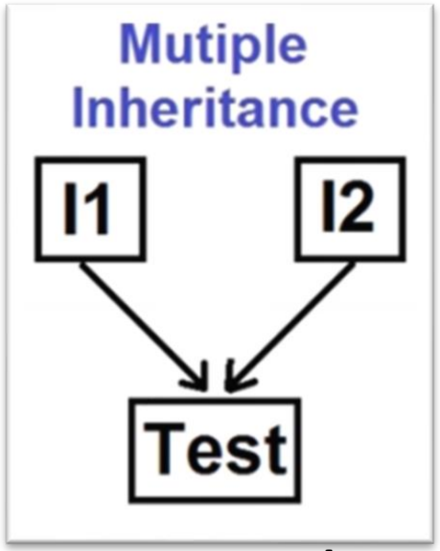
Test.java

```java
interface I1{
	void show();
}
class Test implements I1{
	public void show(){
		System.out.println("1");
	}
	public static void main(String[] args){
		//I1 interfaceobject = new I1();
		Test tobj = new Test();
		tobj.show();
	}
}
```

Select Command Prompt

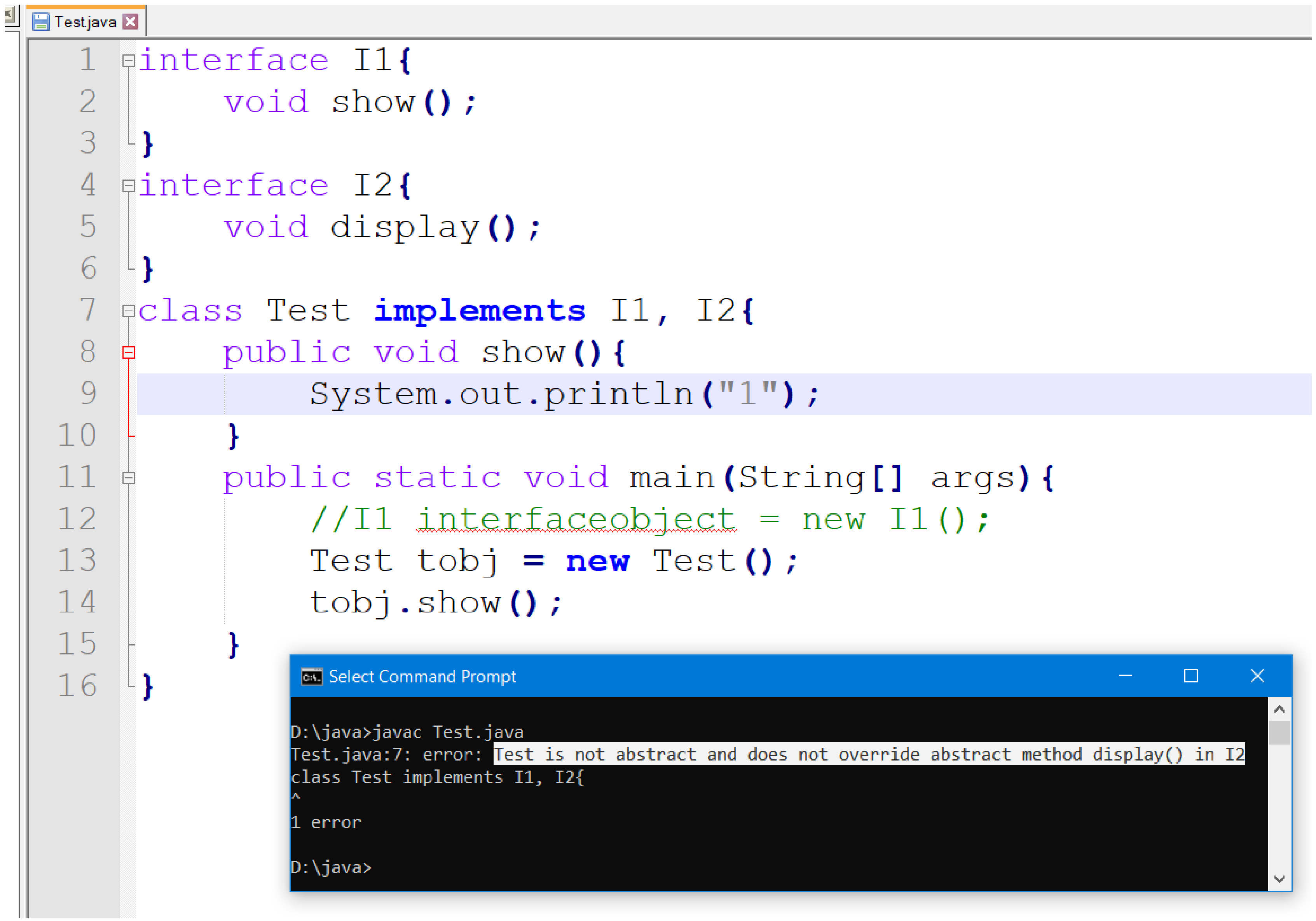
```
Microsoft Windows [Version 10.0.19043.1706]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Rehman Ali>d:

D:\>cd java

D:\java>javac Test.java

D:\java>java Test
1

D:\java>
```

POINT TO NOTE:



Interface is also help us to achieve multi inheritance

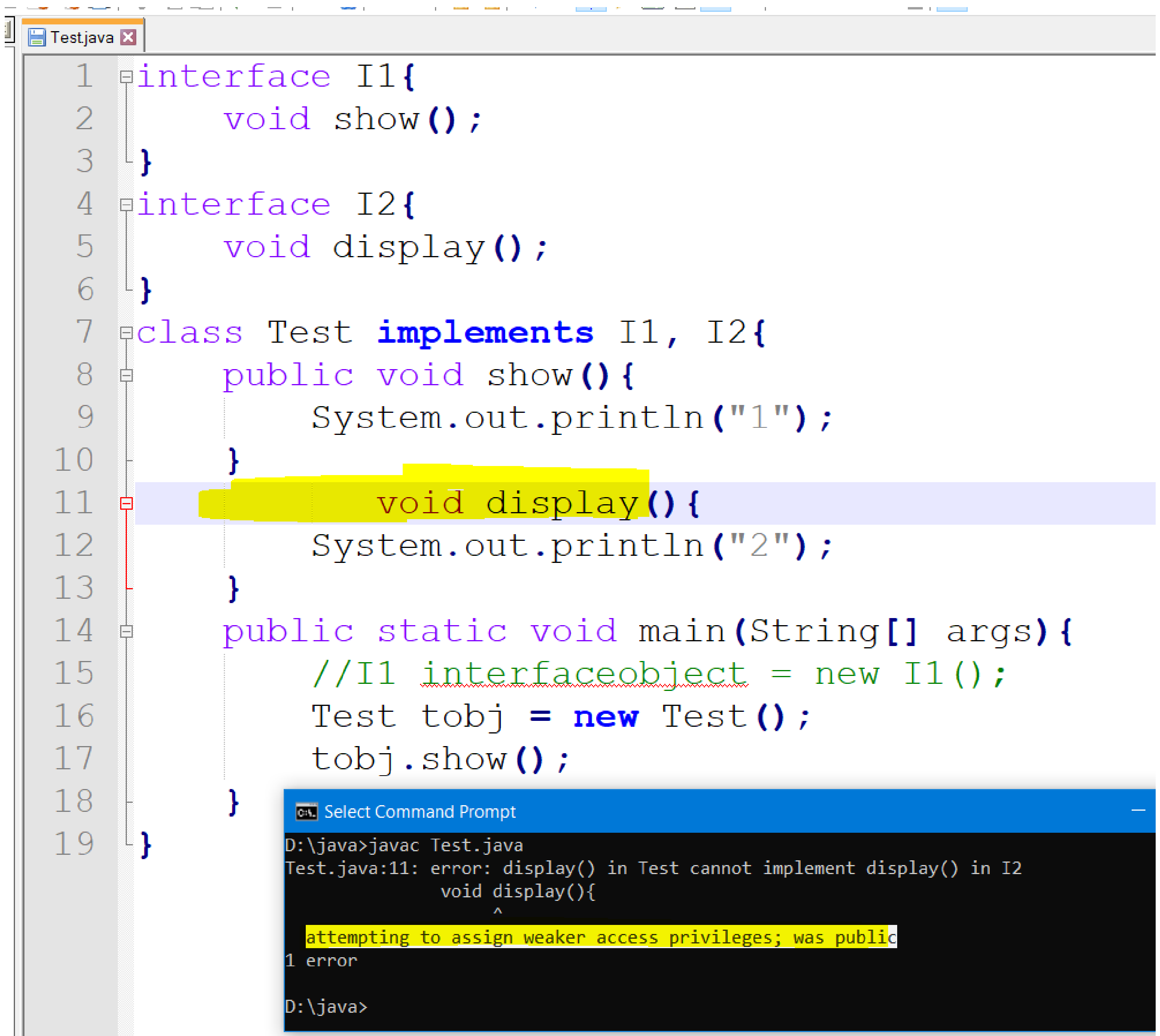
Achieving Multi inheritance using interfaces



```java
interface I1{
	void show();
}
interface I2{
	void display();
}
class Test implements I1, I2{
	public void show(){
		System.out.println("1");
	}
	public static void main(String[] args){
		//I1 interfaceobject = new I1();
		Test tobj = new Test();
		tobj.show();
	}
}
```

```
D:\java>javac Test.java
Test.java:7: error: Test is not abstract and does not override abstract method display() in I2
class Test implements I1, I2{
^
1 error

D:\java>
```

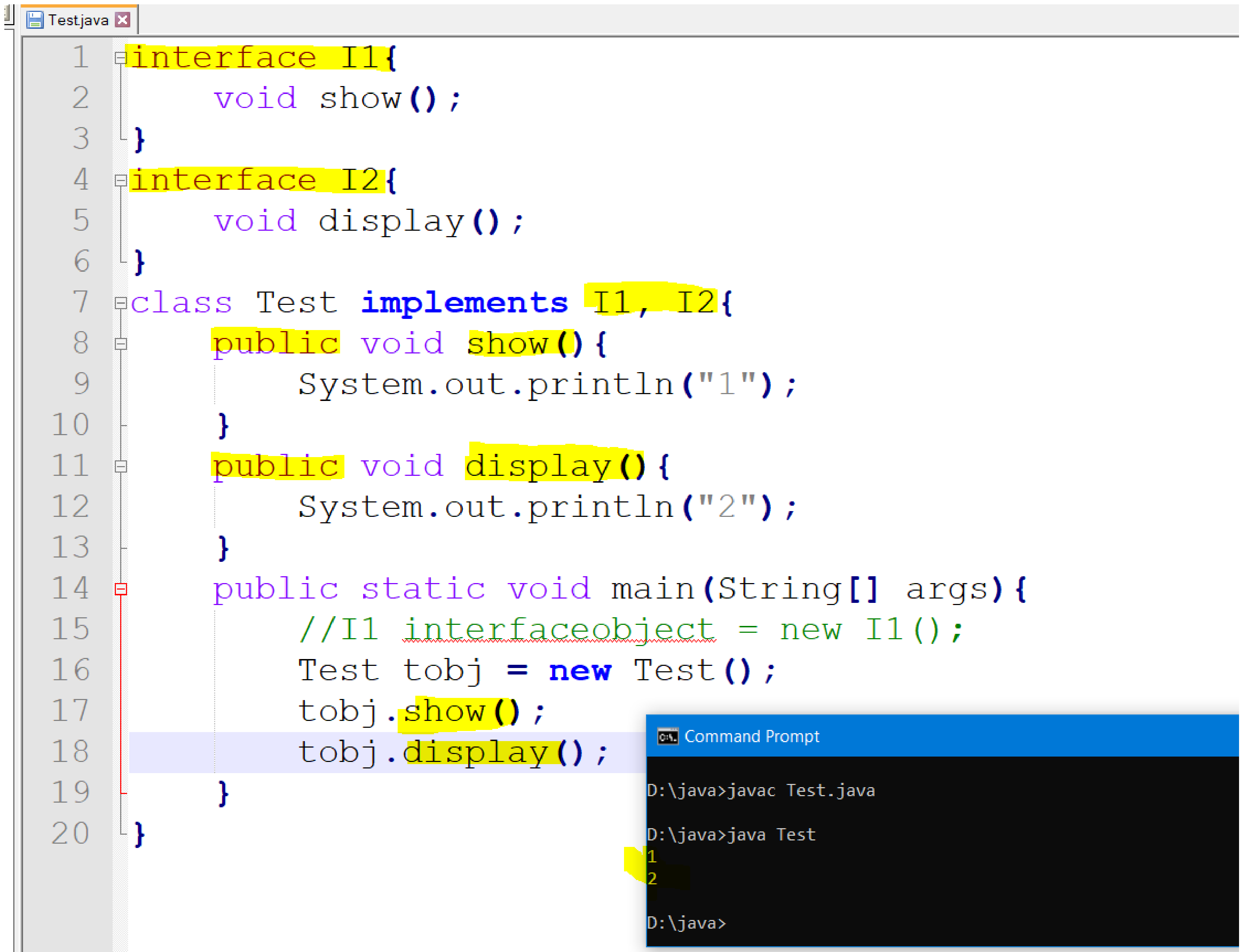
Achieving Multi inheritance using interfaces



```
interface I1{
        void show();
}
interface I2{
        void display();
}
class Test implements I1, I2{
        public void show(){
                System.out.println("1");
        }
                void display(){
                System.out.println("2");
        }
        public static void main(String[] args){
                //I1 interfaceobject = new I1();
                Test tobj = new Test();
                tobj.show();
        }
}
```

```
D:\java>javac Test.java
Test.java:11: error: display() in Test cannot implement display() in I2
              void display(){
                   ^
  attempting to assign weaker access privileges; was public
1 error

D:\java>
```

Achieving Multi inheritance using interfaces

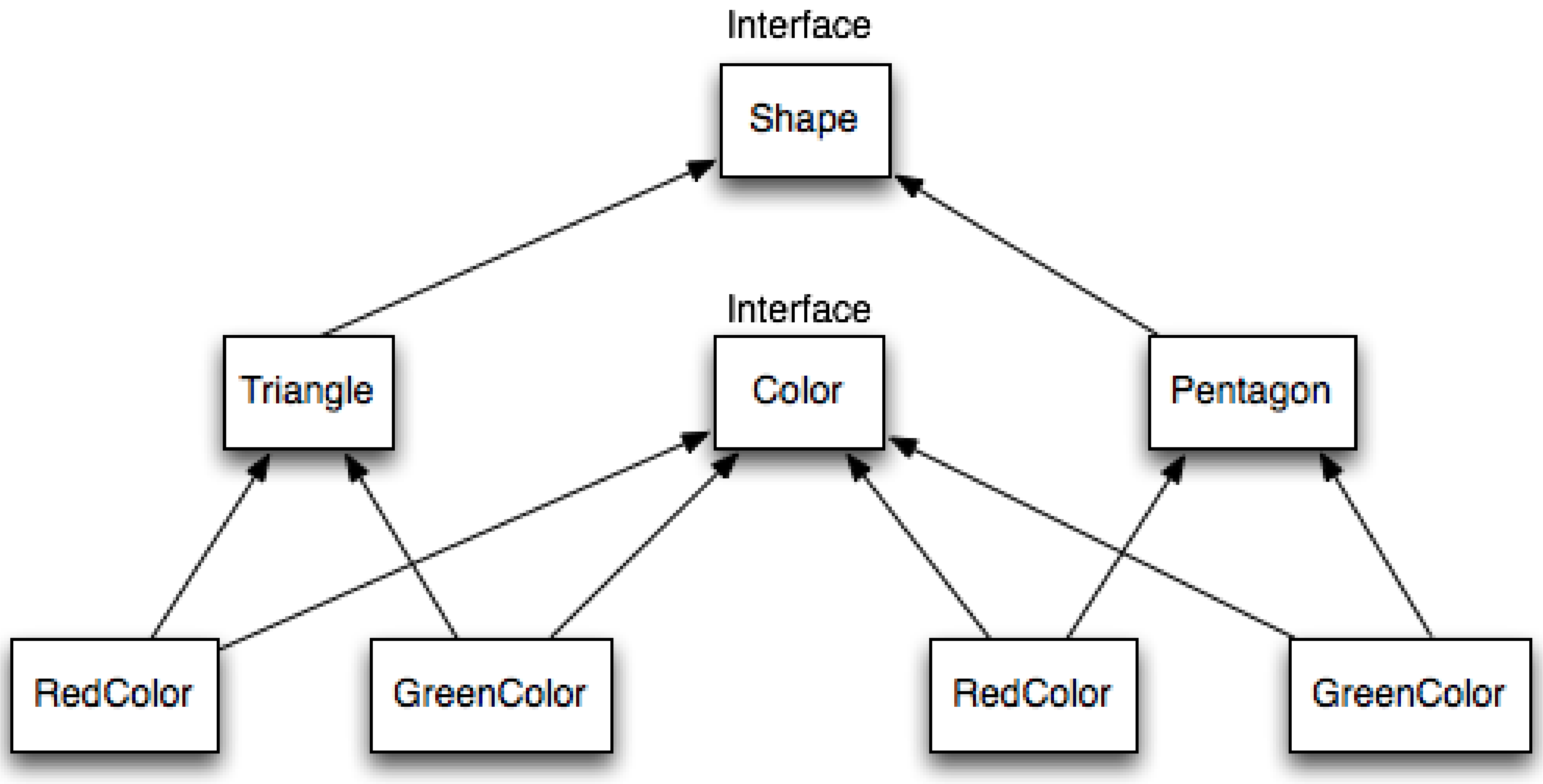


Testjava

```java
interface I1{
        void show();
}
interface I2{
        void display();
}
class Test implements I1, I2{
        public void show(){
                System.out.println("1");
        }
        public void display(){
                System.out.println("2");
        }
        public static void main(String[] args){
                //I1 interfaceobject = new I1();
                Test tobj = new Test();
                tobj.show();
                tobj.display();
        }
}
```

Command Prompt

```
D:\java>javac Test.java

D:\java>java Test
1
2

D:\java>
```


Interface
Shape
Interface
Triangle
Color
Pentagon
RedColor
GreenColor
RedColor
GreenColor

What we have learn from this exercise