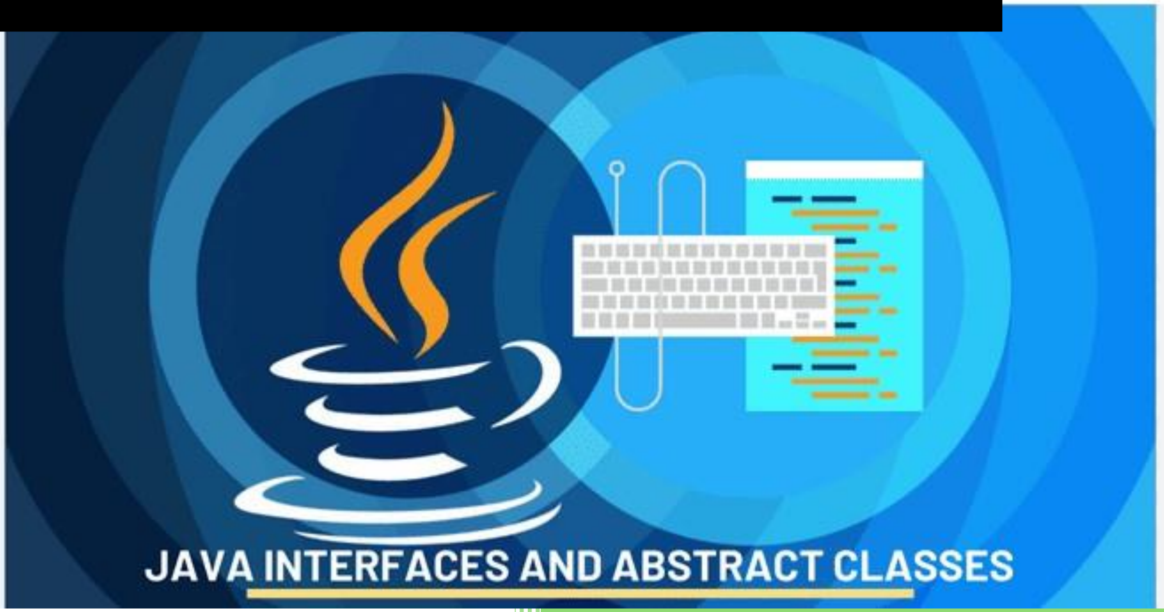
2025

Module 10 Introduction: Abstract Classes and Interfaces



Steve Stylin

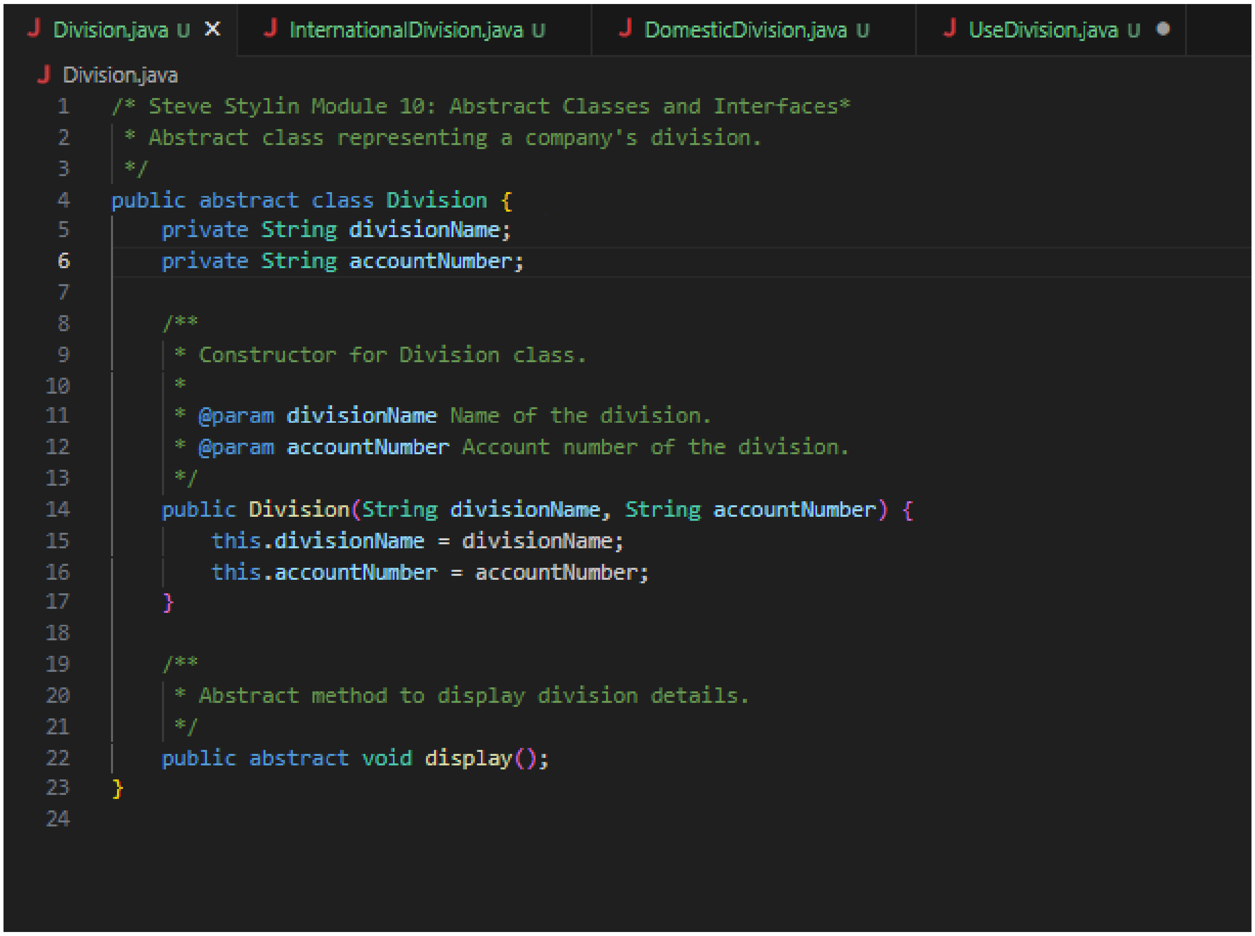
Bellevue University

2/20/2025

The assignment requires the creation of an abstract class named Division. With specific fields and methods, along with two subclasses: InternationalDivision.java and DomesticDivision.java. Additionally, an application named UseDivision.java must be created to instantiate these classes. The question also highlights an error related to access modifiers in Java.

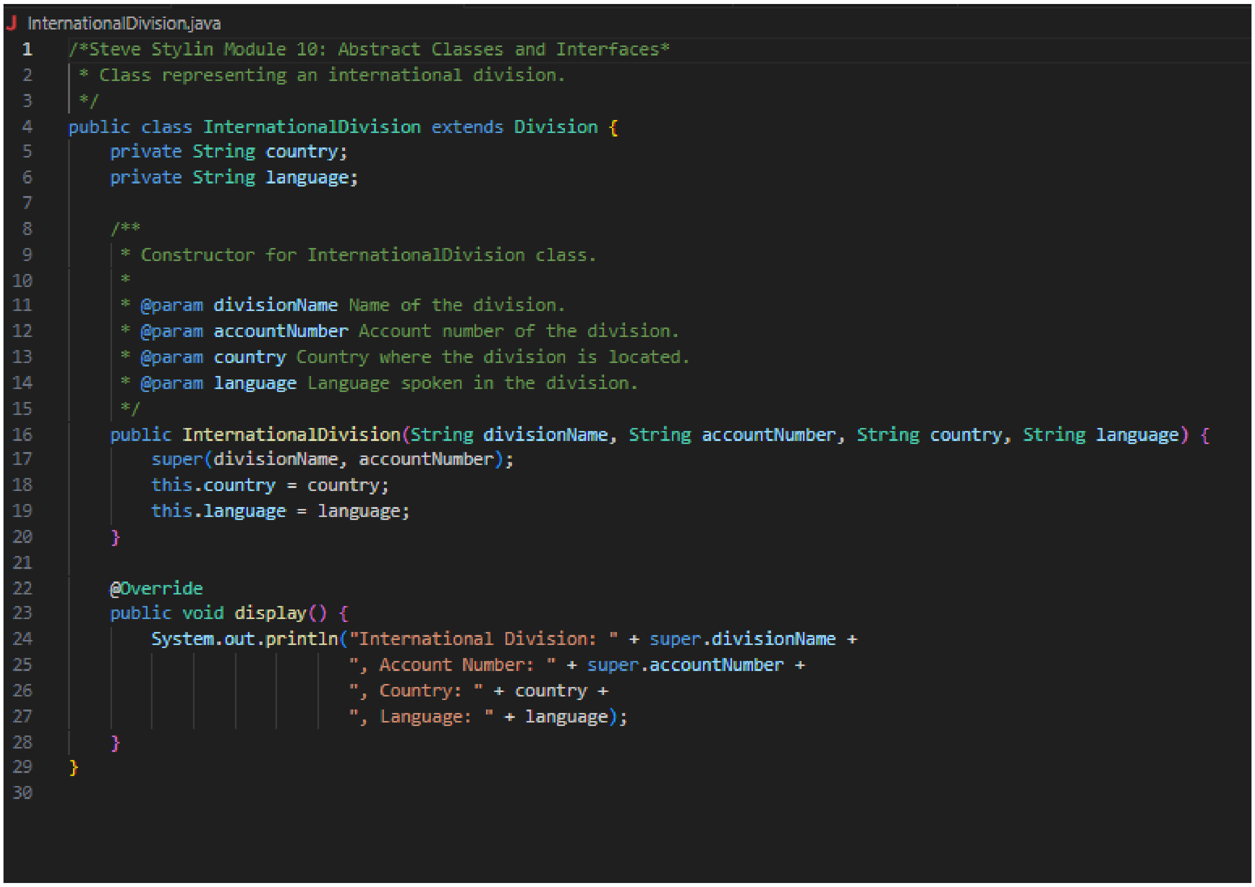
To address the requirements, we will create an abstract class division that contains fields for the division name and account number, as well as an abstract method display(). The subclasses will extend this class and implement the display() method while adding their specific fields

Let's create the Divion.java class:



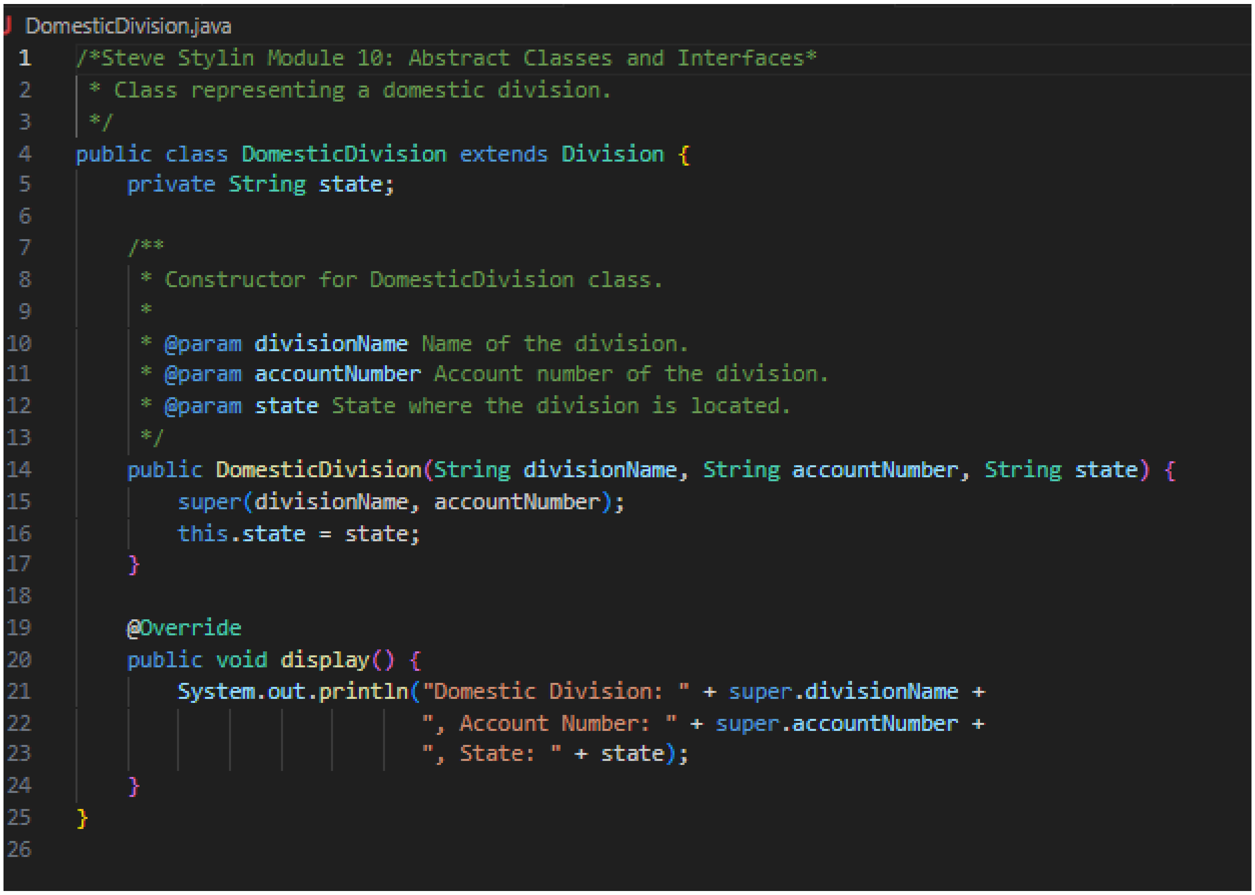
```java
/* Steve Stylin Module 10: Abstract Classes and Interfaces*
 * Abstract class representing a company's division.
 */
public abstract class Division {
    private String divisionName;
    private String accountNumber;

    /**
     * Constructor for Division class.
     *
     * @param divisionName Name of the division.
     * @param accountNumber Account number of the division.
     */
    public Division(String divisionName, String accountNumber) {
        this.divisionName = divisionName;
        this.accountNumber = accountNumber;
    }

    /**
     * Abstract method to display division details.
     */
    public abstract void display();
}
```

We have created the *InternationalDivision.java* to represent an international division.



```java
/*Steve Stylin Module 10: Abstract Classes and Interfaces*
 * Class representing an international division.
 */
public class InternationalDivision extends Division {
    private String country;
    private String language;

    /**
     * Constructor for InternationalDivision class.
     *
     * @param divisionName Name of the division.
     * @param accountNumber Account number of the division.
     * @param country Country where the division is located.
     * @param language Language spoken in the division.
     */
    public InternationalDivision(String divisionName, String accountNumber, String country, String language) {
        super(divisionName, accountNumber);
        this.country = country;
        this.language = language;
    }

    @Override
    public void display() {
        System.out.println("International Division: " + super.divisionName +
                           ", Account Number: " + super.accountNumber +
                           ", Country: " + country +
                           ", Language: " + language);
    }
}
```

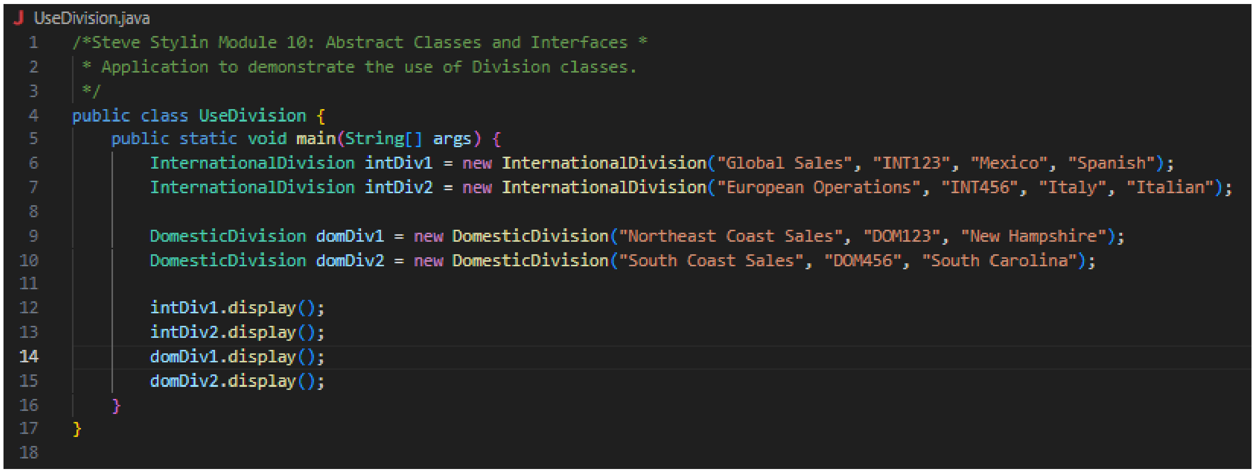
The DomesticDivision.java class is a domestic division.



DomesticDivision.java

```
/*Steve Stylin Module 10: Abstract Classes and Interfaces*
 * Class representing a domestic division.
 */
public class DomesticDivision extends Division {
    private String state;

    /**
     * Constructor for DomesticDivision class.
     *
     * @param divisionName Name of the division.
     * @param accountNumber Account number of the division.
     * @param state State where the division is located.
     */
    public DomesticDivision(String divisionName, String accountNumber, String state) {
        super(divisionName, accountNumber);
        this.state = state;
    }

    @Override
    public void display() {
        System.out.println("Domestic Division: " + super.divisionName +
                           ", Account Number: " + super.accountNumber +
                           ", State: " + state);
    }
}
```

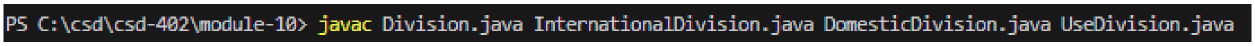
The last step is creating the UseDivision.java Application, which demonstrates the use of Division classes.



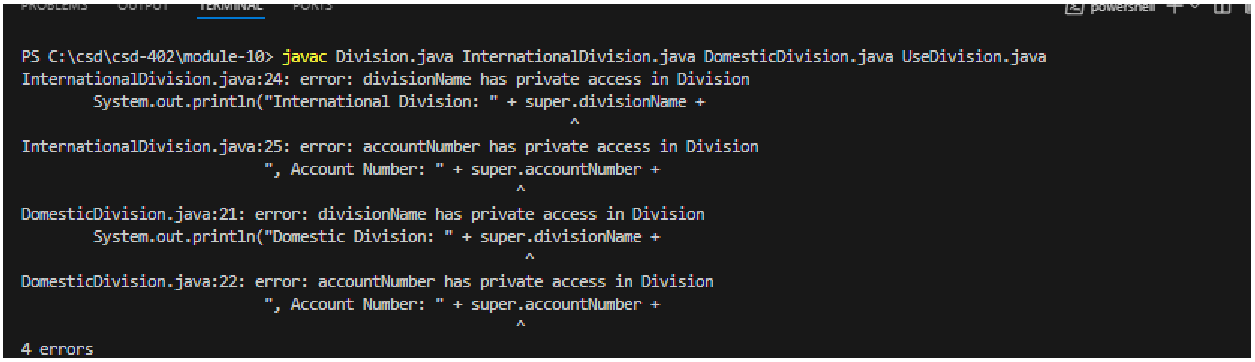
UseDivision.java

```
/*Steve Stylin Module 10: Abstract Classes and Interfaces *
 * Application to demonstrate the use of Division classes.
 */
public class UseDivision {
    public static void main(String[] args) {
        InternationalDivision intDiv1 = new InternationalDivision("Global Sales", "INT123", "Mexico", "Spanish");
        InternationalDivision intDiv2 = new InternationalDivision("European Operations", "INT456", "Italy", "Italian");

        DomesticDivision domDiv1 = new DomesticDivision("Northeast Coast Sales", "DOM123", "New Hampshire");
        DomesticDivision domDiv2 = new DomesticDivision("South Coast Sales", "DOM456", "South Carolina");

        intDiv1.display();
        intDiv2.display();
        domDiv1.display();
        domDiv2.display();
    }
}
```

will compile all the classes in one go. The Java compiler (javac) will resolve dependencies automatically.

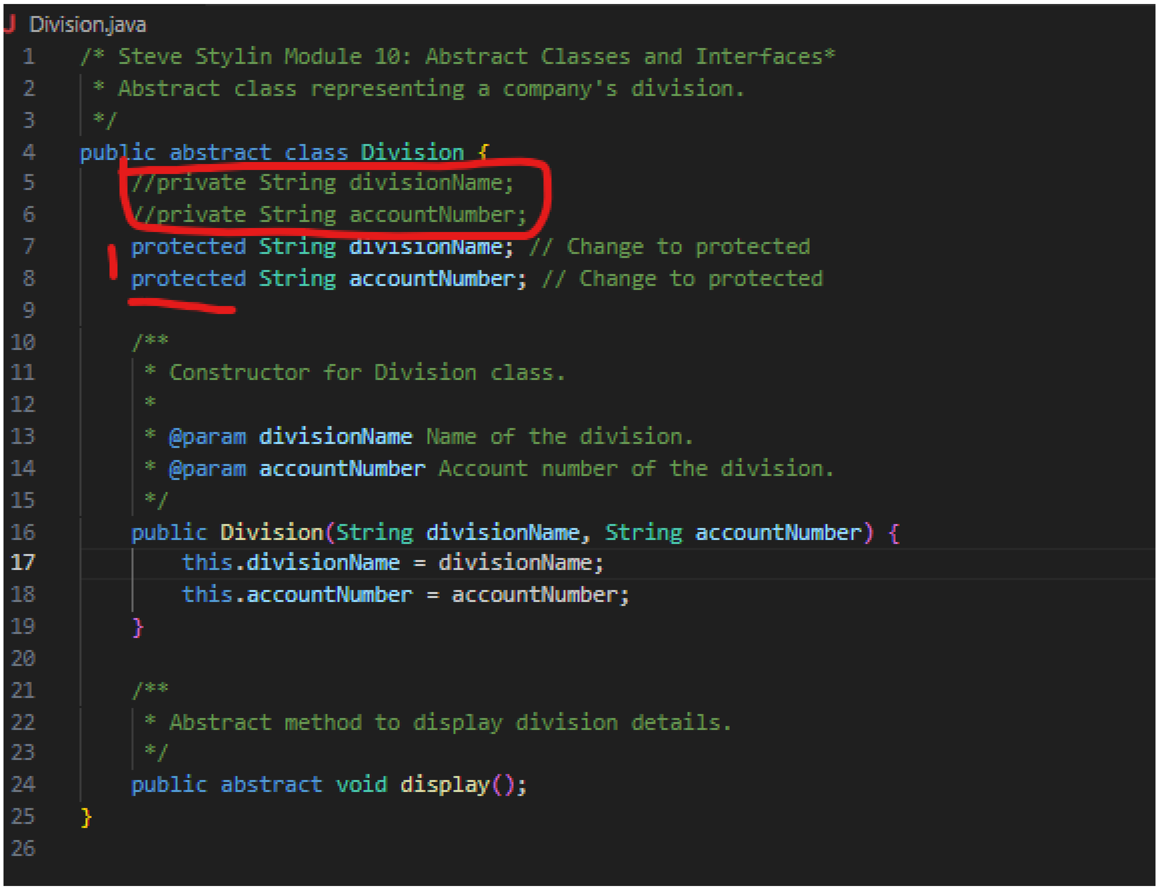


```
PS C:\csd\csd-402\module-10> javac Division.java InternationalDivision.java DomesticDivision.java UseDivision.java
```



```
PS C:\csd\csd-402\module-10> javac Division.java InternationalDivision.java DomesticDivision.java UseDivision.java
InternationalDivision.java:24: error: divisionName has private access in Division
        System.out.println("International Division: " + super.divisionName +
                                                             ^
InternationalDivision.java:25: error: accountNumber has private access in Division
                           ", Account Number: " + super.accountNumber +
                                                       ^
DomesticDivision.java:21: error: divisionName has private access in Division
        System.out.println("Domestic Division: " + super.divisionName +
                                                        ^
DomesticDivision.java:22: error: accountNumber has private access in Division
                           ", Account Number: " + super.accountNumber +
                                                       ^
4 errors
```

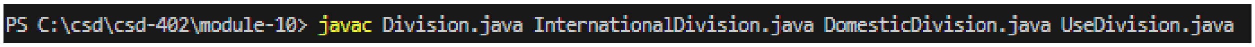
The errors encountered during compilation are due to the access modifiers of the fields in the Division class. The fields divisionName and accountNumber are declared as private, which means they cannot be accessed directly in the subclasses. To resolve this issue, we will change the access modifiers in Division.java from private to protected, allowing subclasses to access them.



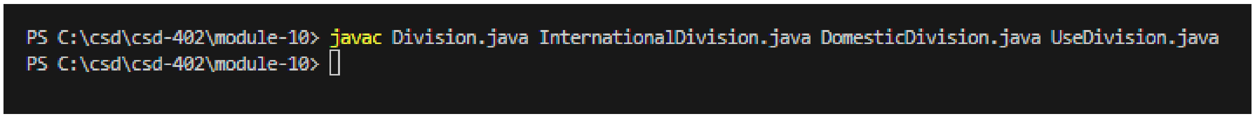
Division.java

```java
/* Steve Stylin Module 10: Abstract Classes and Interfaces*
 * Abstract class representing a company's division.
 */
public abstract class Division {
    //private String divisionName;
    //private String accountNumber;
    protected String divisionName; // Change to protected
    protected String accountNumber; // Change to protected

    /**
     * Constructor for Division class.
     *
     * @param divisionName Name of the division.
     * @param accountNumber Account number of the division.
     */
    public Division(String divisionName, String accountNumber) {
        this.divisionName = divisionName;
        this.accountNumber = accountNumber;
    }

    /**
     * Abstract method to display division details.
     */
    public abstract void display();
}
```

Let us recompile the Java files after modifying Division.java

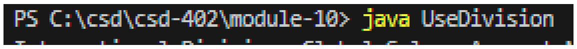


```
PS C:\csd\csd-402\module-10> javac Division.java InternationalDivision.java DomesticDivision.java UseDivision.java
```



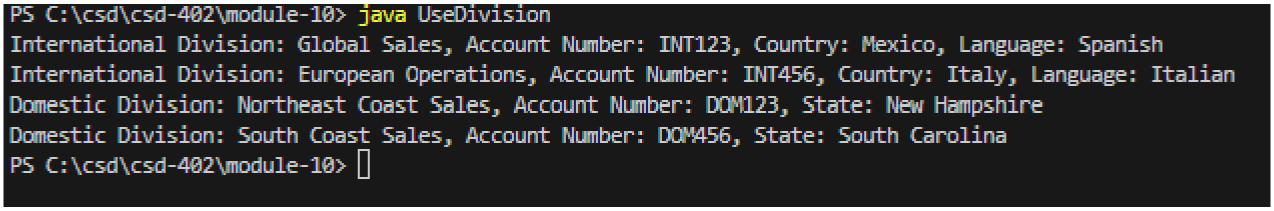
```
PS C:\csd\csd-402\module-10> javac Division.java InternationalDivision.java DomesticDivision.java UseDivision.java
PS C:\csd\csd-402\module-10> ▯
```

The Java files were compiled successfully. Now, we can execute the main method in the UseDivision.java application, creating instances of the divisions and displaying their details.



```
PS C:\csd\csd-402\module-10> java UseDivision
```

Output:



```
PS C:\csd\csd-402\module-10> java UseDivision
International Division: Global Sales, Account Number: INT123, Country: Mexico, Language: Spanish
International Division: European Operations, Account Number: INT456, Country: Italy, Language: Italian
Domestic Division: Northeast Coast Sales, Account Number: DOM123, State: New Hampshire
Domestic Division: South Coast Sales, Account Number: DOM456, State: South Carolina
PS C:\csd\csd-402\module-10> ▯
```