

Contenido

Exercise 1: Using the Java API Documentation.....	2
Exercise 2: Exploring Encapsulation	2
Task 1 – Deleting the Account Class	2
Task 2 – Creating the Account Class	3
Task 3 – Creating the TestAccount2 Class.....	3
Task 4 – Compiling the TestAccount2 Class	4
Task 5 – Running the TestAccount2 Program	4
Exercise 3: Exploring Encapsulation	4
Task 1 – Modifying the Account Class	5
Task 2 – Modifying the TestAccount Class.....	5
Task 3 – Compiling the TestAccount Class	6
Task 4 – Running the TestAccount Program	6
Exercise 4: Creating Java Packages	6
Task 1 – Creating the Java Packages	7
Task 2 – Moving and Modifying the Account Class	7
Task 3 – Moving the TestAccount Class	8
Task 4 – Compiling the TestAccount Class	8
Task 5 – Running the TestAccount Program	8

Exercise 1: Using the Java API Documentation

format

```
public StringBuffer format(Object number,
                          StringBuffer toAppendTo,
                          FieldPosition pos)
```

Formats a number and appends the resulting text to the given string buffer. The number can be of any subclass of `Number`.

This implementation extracts the number's value using `Number.longValue()` for all integral type values that can be converted to `long` without loss of information, including `BigInteger` values with a bit length of less than 64, and `Number.doubleValue()` for all other types. It then calls `format(Long, java.lang.StringBuffer, java.text.FieldPosition)` or `format(double, java.lang.StringBuffer, java.text.FieldPosition)`. This may result in loss of magnitude information and precision for `BigInteger` and `BigDecimal` values.

Specified by:
`format` in class `Format`

Parameters:
`number` - the number to format
`toAppendTo` - the `StringBuffer` to which the formatted text is to be appended
`pos` - On input: an alignment field, if desired. On output: the offsets of the alignment field.

Returns:
the value passed in as `toAppendTo`

Throws:
`IllegalArgumentException` - if `number` is null or not an instance of `Number`.
`NullPointerException` - if `toAppendTo` or `pos` is null
`ArithmeticException` - if rounding is needed with rounding mode being set to `RoundingMode.UNNECESSARY`

See Also:
`FieldPosition`

parseObject

```
public final Object parseObject(String source,
                               ParsePosition pos)
```

Exercise 2: Exploring Encapsulation

Task 1 – Deleting the Account Class

Lab2

- Account.java
- TestAccount.java

File History

Open Timeline

Cut Ctrl+X

Copy Ctrl+C

Copy Path Shift+Alt+C

Copy Relative Path Ctrl+K Ctrl+Shift+C

Copy Remote File URL

Copy Remote File URL From...

Rename... F2

Delete Delete

Run Java

```
getBalance() {
    balance;

    deposit(double amt) {
        balance + amt;

        withdraw(double amt) {
            if (amt >= 0) {
                balance = balance - amt;
            }
        }
    }
}
```

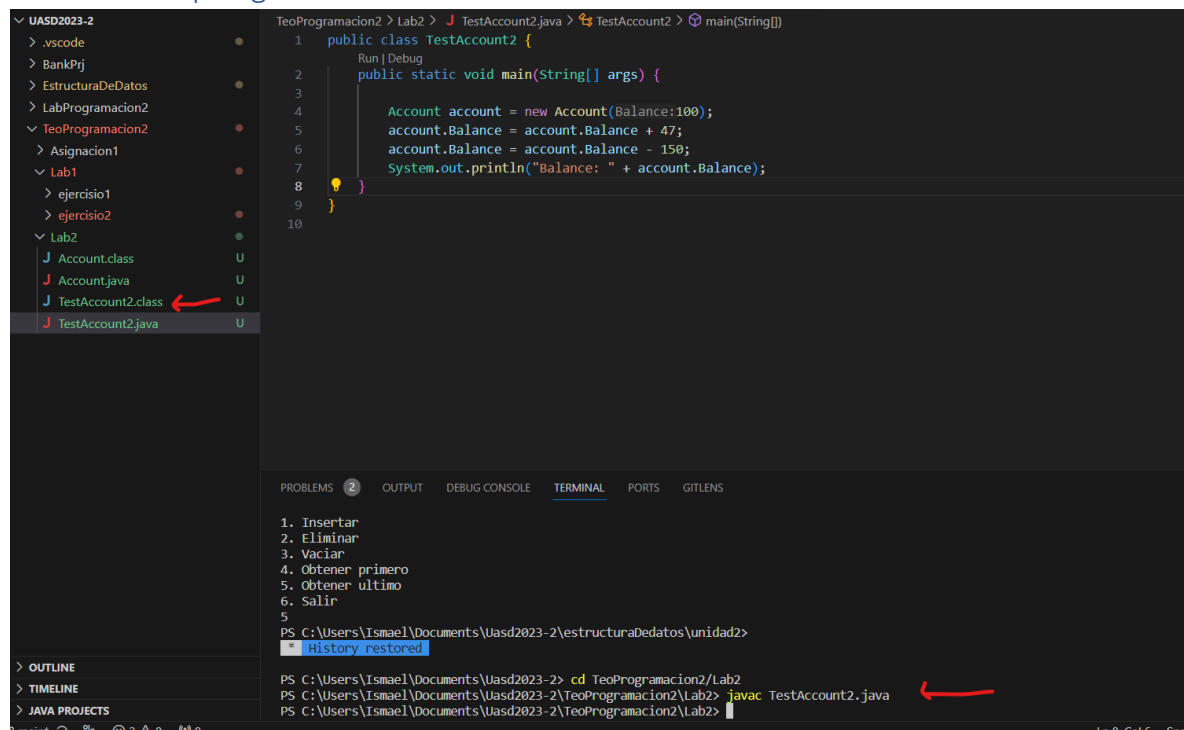
Task 2 – Creating the Account Class

```
TeoProgramacion2 > Lab2 > Account.java > Account > Account(double)
1 public class Account {
2
3     public double Balance;
4
5     public Account( double Balance) {
6         this.Balance = Balance;
7     }
8 }
9
```

Task 3 – Creating the TestAccount2 Class

```
TeoProgramacion2 > Lab2 > TestAccount2.java > TestAccount2 > main(String[])
1 public class TestAccount2 {
2     Run | Debug
3     public static void main(String[] args) {
4
5         Account account = new Account(Balance:100);
6         account.Balance = account.Balance + 47;
7         account.Balance = account.Balance - 150;
8         System.out.println("Balance: " + account.Balance);
9     }
10 }
```

Task 4 – Compiling the TestAccount2 Class

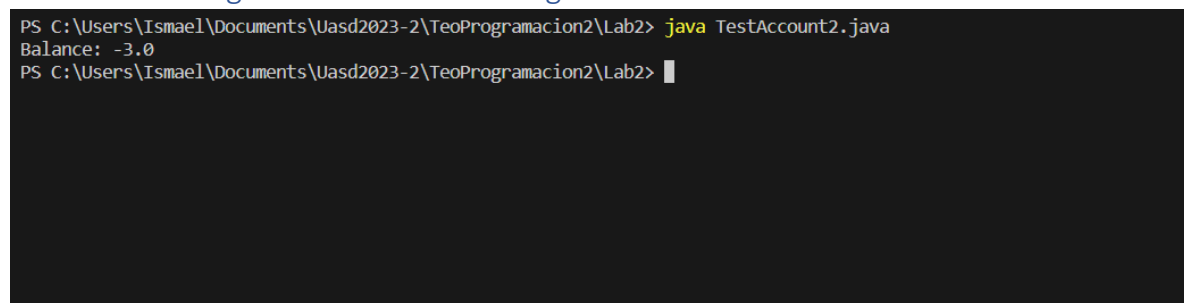


```
TeoProgramacion2 > Lab2 > J TestAccount2.java > TestAccount2 > main(String[])
1 public class TestAccount2 {
2     Run | Debug
3     public static void main(String[] args) {
4
5         Account account = new Account(Balance:100);
6         account.Balance = account.Balance + 47;
7         account.Balance = account.Balance - 150;
8         System.out.println("Balance: " + account.Balance);
9     }
10 }
```

1. Insertar
2. Eliminar
3. Vaciar
4. Obtener primero
5. Obtener ultimo
6. Salir
5
PS C:\Users\Ismael\Documents\Uasd2023-2\estructuraDeDatos\unidad2>
* History restored

PS C:\Users\Ismael\Documents\Uasd2023-2> cd TeoProgramacion2\Lab2
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2> javac TestAccount2.java
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2>

Task 5 – Running the TestAccount2 Program



```
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2> java TestAccount2.java
Balance: -3.0
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2>
```

Exercise 3: Exploring Encapsulation

Task 1 – Modifying the Account Class

TeoProgramacion2 > Lab2 > Account.java > Account > withdraw(double)

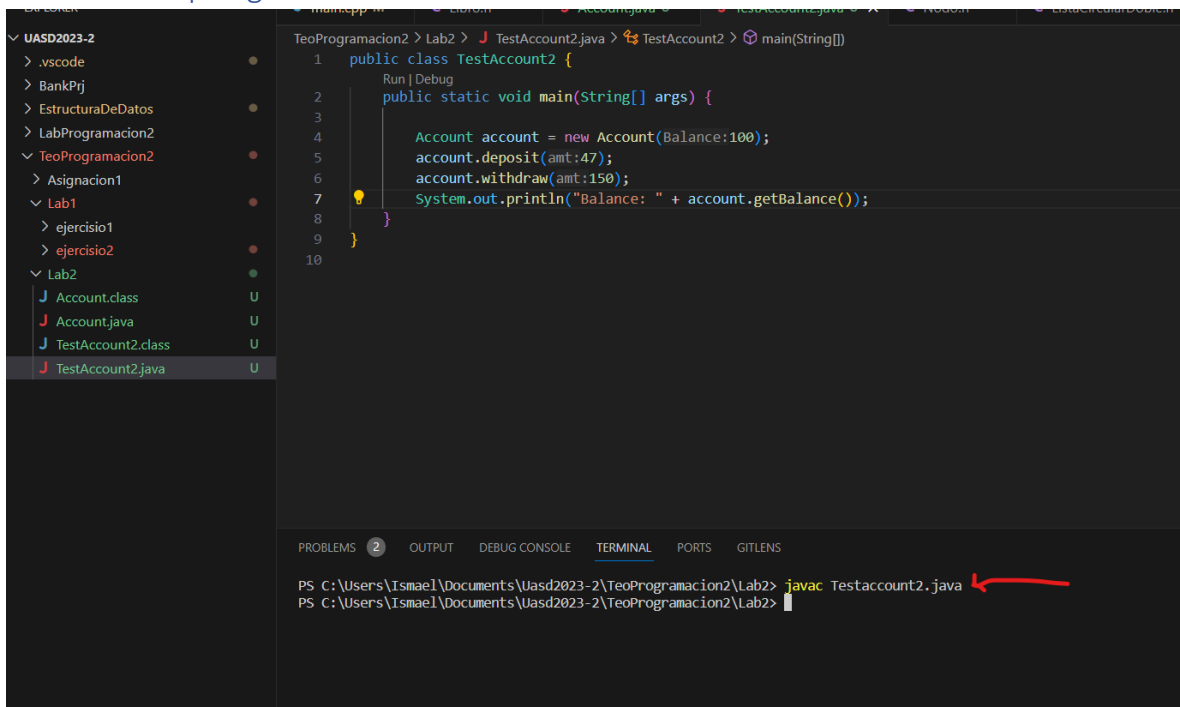
```
1  public class Account {
2
3      private double Balance;
4
5      public Account(double Balance) {
6          this.Balance = Balance;
7      }
8
9      public void deposit(double amt) {
10         Balance = Balance + amt;
11     }
12
13     public void withdraw(double amt) {
14         if (amt <= Balance) {
15             Balance = Balance - amt;
16         }
17     }
18
19     public double getBalance() {
20         return Balance;
21     }
22 }
```

Task 2 – Modifying the TestAccount Class

TeoProgramacion2 > Lab2 > TestAccount2.java > TestAccount2 > main(String[])

```
1  public class TestAccount2 {
2      Run | Debug
3      public static void main(String[] args) {
4
5          Account account = new Account(Balance:100);
6          account.deposit(amt:47);
7          account.withdraw(amt:150);
8          System.out.println("Balance: " + account.getBalance());
9      }
10 }
```

Task 3 – Compiling the TestAccount Class



The screenshot shows the Visual Studio Code interface. On the left, the Explorer panel displays a project structure with folders like 'UASD2023-2', 'BankPrj', 'EstructuraDeDatos', 'LabProgramacion2', 'TeoProgramacion2', 'Asignacion1', 'Lab1', 'ejercicio1', 'ejercicio2', 'Lab2', and files 'Account.class', 'Account.java', 'TestAccount2.class', and 'TestAccount2.java'. The main editor shows the code for 'TestAccount2.java' with the following content:

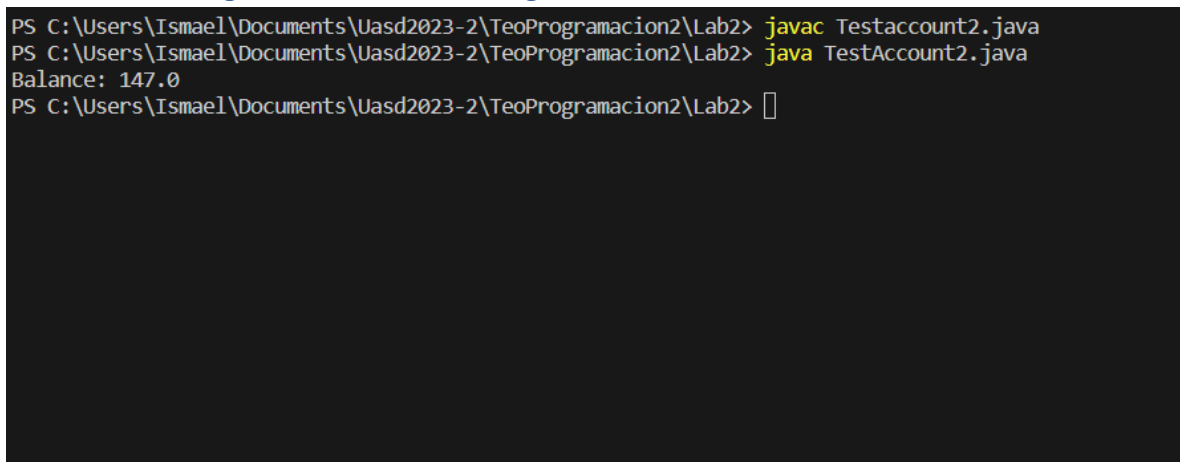
```
1 public class TestAccount2 {  
2     Run | Debug  
3     public static void main(String[] args) {  
4  
5         Account account = new Account(Balance:100);  
6         account.deposit(amt:47);  
7         account.withdraw(amt:150);  
8         System.out.println("Balance: " + account.getBalance());  
9     }  
10 }
```

At the bottom, the TERMINAL panel shows the command prompt with the following commands and output:

```
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2> javac Testaccount2.java  
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2>
```

A red arrow points to the 'javac Testaccount2.java' command in the terminal.

Task 4 – Running the TestAccount Program

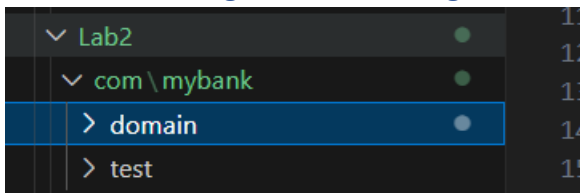


The screenshot shows a terminal window with the following commands and output:

```
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2> javac Testaccount2.java  
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2> java TestAccount2.java  
Balance: 147.0  
PS C:\Users\Ismael\Documents\Uasd2023-2\TeoProgramacion2\Lab2>
```

Exercise 4: Creating Java Packages

Task 1 – Creating the Java Packages



Task 2 – Moving and Modifying the Account Class

```
TeoProgramacion2 > Lab2 > com > mybank > domain > Account.java > ...  
1  package com.mybank.domain;  
2  
3  public class Account {  
4  
5      private double Balance;  
6  
7      public Account(double Balance) {  
8          this.Balance = Balance;  
9      }  
10  
11     public void deposit(double amt) {  
12         Balance = Balance + amt;  
13     }  
14  
15     public void withdraw(double amt) {  
16         if (amt <= Balance) {  
17             Balance = Balance - amt;  
18         }  
19     }  
20  
21     public double getBalance() {  
22         return Balance;  
23     }  
24 }
```

Task 3 – Moving the TestAccount Class

```
TeoProgramacion2 > Lab2 > com > mybank > test > TestAccount2.java > TestAccount2 > n
1  package com.mybank.test;
2
3  import com.mybank.domain.Account;
4
5  public class TestAccount2 {
    Run | Debug
6      public static void main(String[] args) {
7
8          Account account = new Account(Balance:100);
9          account.deposit(amt:47);
10         account.withdraw(amt:150);
11         System.out.println("Balance: " + account.getBalance());
12     }
13 }
14
```

Task 4 – Compiling the TestAccount Class

Task 5 – Running the TestAccount Program

```
Balance: 147.0
PS C:\Users\Ismael\Documents\Uasd2023-2>
```