Arrays:

1. Develop a Java program that declares and initializes an array of integers. Print the elements of the array in reverse order.

```
public static void QuestionOne(){
int a[] = {1,2,3,4,5};
for (int i = a.length - 1; i>=0 ; i--) {
    System.out.println(a[i]);
}
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
5
4
3
2
1
```

2. Implement a Java program that finds the sum and average of elements in an array of floating-point numbers.

```
public static void QuestionTwo(){
double[] numbers = {1.1, 2.2, 3.3, 4.4, 5.5};
double sum = 0;
for(double i:numbers){
    sum+=i;
}
System.out.println(sum);
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
16.5
```

Basics of Classes and Objects:

3.Define a class named BankAccount with attributes accountNumber, balance, and accountHolderName, accountHolderAddress.

```
class BankAccount{
    private String accountNumber;
    private int balance;
    private String accountHolderName;
    private String accountHolderAddress;

    public BankAccount(String accountNumber, int balance, String accountHolderName,
String accountHolderAddress) {
    this.accountNumber = accountNumber;
    this.balance = balance;
    this.accountHolderName = accountHolderName;
    this.accountHolderAddress = accountHolderAddress;
    }
```

```
    public BankAccount() {
    this.accountNumber = "000";
    this.balance = 0;
    this.accountHolderName = "Default-Account";
    this.accountHolderAddress = "Default-Street";
    }

    public void depositMoney(int toAdd){
    this.balance += toAdd;
    }
    public void withDrawMoney(int toSub){
    this.balance -= toSub;
    }
    public int getBalance(){
    return this.balance;
    }
}
```

4.Create an object of this class and initialize its attributes.

```
    public static void QuestionThree(){
    BankAccount account = new BankAccount("12345",1000,"Someone","Kathmandu");
    BankAccount account2 = new BankAccount();
    account.depositMoney(200);
    account.withDrawMoney(100);
    System.out.println(account.getBalance());
    }
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
1100
```

Methods:

5. Create a method, **depositMoney()** in the BankAccount class to deposit money. Implement another method, **withdrawMoney()** to withdraw money. (The current balance should also be printed).

```
class BankAccount{
    private String accountNumber;
    private int balance;
    private String accountHolderName;
    private String accountHolderAddress;

    public BankAccount(String accountNumber, int balance, String accountHolderName,
String accountHolderAddress) {
    this.accountNumber = accountNumber;
    this.balance = balance;
    this.accountHolderName = accountHolderName;
    this.accountHolderAddress = accountHolderAddress;
```

```
    }

    public BankAccount() {
    this.accountNumber = "000";
    this.balance = 0;
    this.accountHolderName = "Default-Account";
    this.accountHolderAddress = "Default-Street";
    }

    public void depositMoney(int toAdd){
    this.balance += toAdd;
    System.out.println("Current Balance: "+this.balance);
    }
    public void withDrawMoney(int toSub){
    this.balance -= toSub;
    System.out.println("Current Balance: "+this.balance);
    }
    public int getBalance(){
    return this.balance;
    }
}
```

```
    public static void QuestionThree(){
    BankAccount account = new BankAccount("12345",1000,"Someone","Kathmandu");
    BankAccount account2 = new BankAccount();
    account.depositMoney(200);
    account.withDrawMoney(100);
    System.out.println(account.getBalance());
    }
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
Current Balance: 1200
Current Balance: 1100
1100
```

6. Create a class Lamp with attributes **isOn** to store boolean value. Also create a method **turnOn()** to turn on the light, and **turnOff()** to turn off the light and print the on status of the light.

```
class Lamp{
    private boolean IsOn;
    public Lamp(){
    this.IsOn = false;
    }
    public void turnOn(){
    this.IsOn = true;
    System.out.println(this.IsOn);
    }
    public void turnOff(){
    this.IsOn = false;
```

```
    System.out.println(this.IsOn);
    }
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
true
false
```

Constructors:

7. Implement a parameterized constructor for the BankAccount class that initializes the account attributes. Create an object using this constructor.

```
class BankAccount{
    private String accountNumber;
    private int balance;
    private String accountHolderName;
    private String accountHolderAddress;

    public BankAccount(String accountNumber, int balance, String accountHolderName,
String accountHolderAddress) {
    this.accountNumber = accountNumber;
    this.balance = balance;
    this.accountHolderName = accountHolderName;
    this.accountHolderAddress = accountHolderAddress;
    }

    public BankAccount() {
    this.accountNumber = "000";
    this.balance = 0;
    this.accountHolderName = "Default-Account";
    this.accountHolderAddress = "Default-Street";
    }

    public void depositMoney(int toAdd){
    this.balance += toAdd;
    System.out.println("Current Balance: "+this.balance);
    }
    public void withDrawMoney(int toSub){
    this.balance -= toSub;
    System.out.println("Current Balance: "+this.balance);
    }
    public int getBalance(){
    return this.balance;
    }
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
Current Balance: 1200
Current Balance: 1100
1100
```

8. Implement a no-argument constructor that prints out “**User created!**” as soon as the instance of the user is created.

```
class BankAccount{
    private String accountNumber;
    private int balance;
    private String accountHolderName;
    private String accountHolderAddress;

    public BankAccount(String accountNumber, int balance, String accountHolderName,
String accountHolderAddress) {
    this.accountNumber = accountNumber;
    this.balance = balance;
    this.accountHolderName = accountHolderName;
    this.accountHolderAddress = accountHolderAddress;
    }

    public BankAccount() {
     System.out.println(“User Created!”);
    }

    public void depositMoney(int toAdd){
    this.balance += toAdd;
    System.out.println("Current Balance: "+this.balance);
    }
    public void withDrawMoney(int toSub){
    this.balance -= toSub;
    System.out.println("Current Balance: "+this.balance);
    }
    public int getBalance(){
    return this.balance;
    }
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
user Created
Current Balance: 1200
Current Balance: 1100
1100
```

Constructor Overloading:

9. Create a class named ,”**Box**” with attributes **width**, **height**, and **depth.** Create multiple constructors for handling following object declarations. Also declare a method **getVolume()** that prints the volume of the declared:

   a. For a cube, declare a constructor to take length only.

b. For a cuboid, declare a constructor to take length, breadth, and height.

c. For no parameter, declare a no-argument constructor that sets **length = 10**, **breadth = 8**, and **height = 12**.

```
class Box {
    private int width;
    private int length;
    private int height;

    public Box(int width, int length ,int height){
    this.width = width;
    this.length = length;
    this.height = height;
    }

    public Box(int length){
    this.length = length;
    }

    public Box(){
    this.length= 10;
    this.width = 8;
    this.height= 12;
    }

    public void getVolume(){
    if (this.width == 0 && this.height == 0) {
        System.out.println(this.length * this.length * this.length);
    }else{
        System.out.println(this.length * this.width * this.height);
    }
    }
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
1000
1000
960
```

Access Modifiers:

10.Set the balance attribute in the BankAccount class as private. Provide public getter methods for the balance.

```
class BankAccount{
    private String accountNumber;
    private int balance;
    private String accountHolderName;
    private String accountHolderAddress;
```

```
    public BankAccount(String accountNumber, int balance, String accountHolderName,
String accountHolderAddress) {
    this.accountNumber = accountNumber;
    this.balance = balance;
    this.accountHolderName = accountHolderName;
    this.accountHolderAddress = accountHolderAddress;
    }

    public BankAccount() {
    this.accountNumber = "000";
    this.balance = 0;
    this.accountHolderName = "Default-Account";
    this.accountHolderAddress = "Default-Street";
    }

    public void depositMoney(int toAdd){
    this.balance += toAdd;
    System.out.println("Current Balance: "+this.balance);
    }
    public void withDrawMoney(int toSub){
    this.balance -= toSub;
    System.out.println("Current Balance: "+this.balance);
    }
    public int getBalance(){
    return this.balance;
    }
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
Current Balance: 1200
Current Balance: 1100
1100
```

Encapsulation:

11.Create a class Address with private attributes street, city, and zipCode. Use encapsulation and provide getter and setter methods.

```
class Address{
    private String street;
    private String city;
    private String zip;

    public Address(String street, String city, String zip){
    this.street = street;
    this.city = city;
    this.zip = zip;
```

```
    }

    public String[] getStuff(){
    String holder[] = {this.street,this.city,this.zip};
    return holder;
    }

    public void setStuff(String street,String city, String zip){
    this.street = street;
    this.city = city;
    this.zip = zip;
    }
}
```

```
    public static void QuestionTen(){
    Address addr = new Address("something","Kathmandu","44600");
    for(String i: addr.getStuff()){
        System.out.println(i);
    }
    }
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
something
Kathmandu
44600
```

Combining Concepts:

12.Create a class Customer with private attributes customerId, name, and a BankAccount attribute. Implement a parameterized constructor and encapsulate the attributes. Provide getter method. Instantiate multiple Customer objects with different values and demonstrate the use of getters and setters.

```
class Customer{
    private String customerId;
    private String name;
    private String bankAccount;
    public Customer(String Id, String name, String bankAccount){
    this.customerId = Id;
    this.name = name;
    this.bankAccount = bankAccount;
    }
    public String[] getStuff(){
    String holder[] = {this.customerId,this.name,this.bankAccount};
    return holder;
    }
    public void setStuff(String customerId,String name, String bankAccount){
    this.customerId = customerId;
```

```
    this.name = name;
    this.bankAccount = bankAccount;
    }
}
```

```
public static void QuestionTwelve(){
    Customer cust1 = new Customer("1","Someone0","12346");
    Customer cust2 = new Customer("2","Someone1","12347");
    for(String i: cust1.getStuff()){
        System.out.println(i);
    }
    cust1.setStuff("2","ChangedSomeone","342423");
    for(String i: cust1.getStuff()){
        System.out.println(i);
    }
    }
```

Constructors Overloading:

13.Implement multiple constructors for the BankAccount class with different parameter sets. Use constructor overloading to create objects with different initialization scenarios.

```
class BankAccount{
    private String accountNumber;
    private int balance;
    private String accountHolderName;
    private String accountHolderAddress;

    public BankAccount(String accountNumber, int balance, String accountHolderName,
String accountHolderAddress) {
    this.accountNumber = accountNumber;
    this.balance = balance;
    this.accountHolderName = accountHolderName;
    this.accountHolderAddress = accountHolderAddress;
    }

    public BankAccount() {
    this.accountNumber = "000";
    this.balance = 0;
    this.accountHolderName = "Default-Account";
    this.accountHolderAddress = "Default-Street";
    }

    public void depositMoney(int toAdd){
    this.balance += toAdd;
```

```
    System.out.println("Current Balance: "+this.balance);
    }
    public void withDrawMoney(int toSub){
    this.balance -= toSub;
    System.out.println("Current Balance: "+this.balance);
    }
    public int getBalance(){
    return this.balance;
    }
}
```

```
[nathan@archlinux ugabgua]$ javac Main.java && java Main
Current Balance: 1200
Current Balance: 1100
1100
```