## Class Diagram

```java
public class BankAccount {
    public BankAccount(Customer customer, long id) {
        // field initializations
    }
    public void withdraw(double amount) {
        // implementation
    }
    public void deposit(double amount) {
        // implementation
    }
    public void transferTo(double amount, BankAccount otherAccount) {
        // implementation
    }
    public double getBalance() {
        // implementation
    }
    public Customer getOwner() {
        // implementation
    }
    public double getNumber() {
        // implementation
    }
}
```

## Customer

```java
public class Customer {
    public Customer(String name, String id) {
        // field initializations
    }
    public String getName() {
        // implementation
    }
    public String getID() {
        // implementation
    }
    private String name;
    private String id;
}
```

## Toy Bank Program

```java
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");
        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);
        System.out.println("account 1 has "+ account1.getBalance());
        System.out.println("account 2 has "+ account2.getBalance());
    }
}
```

## Object Diagram
Toy Bank Program

```java
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");

        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);

        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);

        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

Object Diagram

```java
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");

        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);

        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);

        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}
```

Message Sequence Chart

```java
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");

        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);

        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);

        System.out.println("account1 has " + account1.getBalance());
        System.out.println("account2 has " + account2.getBalance());
    }
}```
```java
public class Bank {
    public static void main(String[] args) {
        Customer customer1 = new Customer("Avi Cohen", "025285244");
        Customer customer2 = new Customer("Rita Stein", "024847638");

        BankAccount account1 = new BankAccount(customer1, 1234);
        BankAccount account2 = new BankAccount(customer2, 5678);
        BankAccount account3 = new BankAccount(customer2, 2984);

        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(100, account3);
        account2.withdraw(300);

        System.out.println("account 1 has "+ account1.getBalance());
        System.out.println("account 2 has "+ account2.getBalance());
    }
}
```

---

**Output**

```
account 1 has 900.0
account 2 has 200.0
```
public interface Playable {
    public void play();
}

public class VideoClip implements Playable {
    @Override
    public void play() {
        // render video, play the clip on screen...
    }
    // does complicated stuff related to video formats...
}

public class MP3Song implements Playable {
    @Override
    public void play() {
        // audio codec calculations, play the song...
    }
    // does complicated stuff related to MP3 format...
}

public class VideoClip {
    public void play();
    // render video, play the clip on screen...
    // does complicated stuff related to video formats...
}

public class MP3Song {
    public void play();
    // audio codec calculations, play the song...
    // does complicated stuff related to MP3 format...
}

public class Player {
    public void playVideos(VideoClip[] clips) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(clips));
            for (VideoClip videoClip : clips)
                videoClip.play();
        } while (repeat);
    }
}

public class VideoClip {
    public void play();
    // render video, play the clip on screen...
    // does complicated stuff related to video formats...
}

public class MP3Song {
    public void play();
    // audio codec calculations, play the song...
    // does complicated stuff related to MP3 format...
}

public class Player {
    public void playSongs(MP3Song[] songs) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(songs));
            for (MP3Song song : songs)
                song.play();
        } while (repeat);
    }
}

public class VideoClip {
    public void play();
    // render video, play the clip on screen...
    // does complicated stuff related to video formats...
}

public class MP3Song {
    public void play();
    // audio codec calculations, play the song...
    // does complicated stuff related to MP3 format...
}

public class Player {
    public void playVideos(VideoClip[] clips) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(clips));
            for (VideoClip videoClip : clips)
                videoClip.play();
        } while (repeat);
    }
}

public class VideoClip implements Playable {
    @Override
    public void play() {
        // render video, play the clip on screen...
    }
    // does complicated stuff related to video formats...
}

public class MP3Song implements Playable {
    @Override
    public void play() {
        // audio codec calculations, play the song...
    }
    // does complicated stuff related to MP3 format...
}

public class Player {
    public void playVideos(VideoClip[] clips) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(clips));
            for (VideoClip videoClip : clips)
                videoClip.play();
        } while (repeat);
    }
}

public class VideoClip {
    public void play();
    // render video, play the clip on screen...
    // does complicated stuff related to video formats...
}

public class MP3Song {
    public void play();
    // audio codec calculations, play the song...
    // does complicated stuff related to MP3 format...
}

public class Player {
    public void playSongs(MP3Song[] songs) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(songs));
            for (MP3Song song : songs)
                song.play();
        } while (repeat);
    }
}

public class VideoClip implements Playable {
    @Override
    public void play() {
        // render video, play the clip on screen...
    }
    // does complicated stuff related to video formats...
}

public class MP3Song implements Playable {
    @Override
    public void play() {
        // audio codec calculations, play the song...
    }
    // does complicated stuff related to MP3 format...
}

public class Player {
    public void playVideos(VideoClip[] clips) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(clips));
            for (VideoClip videoClip : clips)
                videoClip.play();
        } while (repeat);
    }
}

public class VideoClip {
    public void play();
    // render video, play the clip on screen...
    // does complicated stuff related to video formats...
}

public class MP3Song {
    public void play();
    // audio codec calculations, play the song...
    // does complicated stuff related to MP3 format...
}

public class Player {
    public void playSongs(MP3Song[] songs) {
        do {
            if (shuffle)
                Collections.shuffle(Arrays.asList(songs));
            for (MP3Song song : songs)
                song.play();
        } while (repeat);
    }
}
operations on bits

- unary bitwise complement
- signed left shift
- signed right shift
- unsigned right shift
- bitwise AND
- bitwise XOR
- bitwise OR

examples

32-bit int

<table>
<thead>
<tr>
<th>value</th>
<th>binary representation</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>00000000000000000000000000000011</td>
</tr>
<tr>
<td>-3</td>
<td>11111111111111111111111111111111</td>
</tr>
<tr>
<td>-3 &lt;&lt; 2</td>
<td>00000000000000000000000000001000</td>
</tr>
<tr>
<td>-3 &gt;&gt; 1</td>
<td>11111111111111111111111111111010</td>
</tr>
<tr>
<td>-3 &gt;&gt;&gt; 1</td>
<td>11111111111111111111111111111010</td>
</tr>
</tbody>
</table>

what is \( i \& 3 \)?

what is \( i \& 0 \times F \)?

the ones and zeros

representations of numbers in two's complement format:

-2
-1
0
1

2

shifts

- two's complement (or two's complement) is obtained by reversing the bits and adding 1


-11111111111111111111111111111111

-15

what is the word order of an 8-bit integer?