## Example 1

A contract between supplier and customer:

- **Precondition**:
  ```
  * precondition: arr != null
  * postcondition: 
    If ((arr.length==0) || (arr contains only NaNs))
    returns Infinity.
    Otherwise, returns the minimal value in arr.
  ```

```java
public static double min2(double[] arr) {
    double m = Double.POSITIVE_INFINITY;
    for (double x : arr)
        m = (x < m ? x : m);
    return m;
}
```
(precondition 4 (ללא propósito כ- NaN)

```java
public static double min4(double[] arr) {
    double m = Double.POSITIVE_INFINITY;
    for (double x : arr) {
        if (Double.isNaN(x))
            return x;
        m = (x < m ? x : m);
    }
    return m;
}
```

((domino 3 (טיווח שעון ב- NaN)

```java
public static double min3(double[] arr) {
    double m = Double.POSITIVE_INFINITY;
    for (double x : arr) {
        if (arr == null || arr.length == 0)
            return Double.NaN;
    }
    return m;
}
```

((domino 5 (ללא propósito כ- NaN)

```java
public static double min5(double[] arr) {
    if (arr == null)
        return 0;
    double m = Double.POSITIVE_INFINITY;
    for (double x : arr)
        m = (x < m ? x : m);
    return m;
}
```

(פקודת ה- 'להפקיד'

```java
public static void transfer(double amount, BankAccount from, BankAccount to) {
    from.withdraw(amount);
    to.deposit(amount);
}
```

(פקודת ה- 'למשוך'

```java
public void withdraw(double amount) {
    balance -= amount;
}
```

(nEnterprise: ATM (מקהל טosex) - מתוכנת

```java
public static void main(String[] args) {
    BankAccount atm = new BankAccount();
    atm.deposit(100.0);
    atm.withdraw(50.0);
    System.out.println(atm.getBalance());
}
```

(Enterprise: Banking (מקהל טosex) - מתוכנת

```java
public void transfer(double amount, BankAccount from, BankAccount to) {
    from.withdraw(amount);
    to.deposit(amount);
}
```
The bank account class:

```java
public class BankAccount {
    // Constructor
    public BankAccount(Customer customer, long id) {
        accountNumber = id;
        owner = customer;
    }
    // Methods
    public void withdraw(double amount) {
        double balance = getBalance();
        if (balance < amount) {
            throw new IllegalArgumentException("Insufficient balance");
        }
        balance -= amount;
        setBalance(balance);
    }
    public void deposit(double amount) {
        double balance = getBalance();
        balance += amount;
        setBalance(balance);
    }
    public double getBalance() {
        return balance;
    }
    public Customer getOwner() {
        return owner;
    }
}
```

**Class Invariant**

- The invariant for the `BankAccount` class is:
  - `getBalance() >= 0`
  - `getAccountNumber() > 0`
  - `getOwner() != null`

**Public Methods**

- `withdraw(double amount)`: Withdraws the specified amount from the account.
- `deposit(double amount)`: Deposits the specified amount into the account.
- `getBalance()`: Returns the current balance of the account.
- `getOwner()`: Returns the owner of the account.

**Private Methods**

- `setBalance(double balance)`: Sets the balance of the account.

**System Diagram**

- The diagram illustrates the relationships between the `BankAccount` class and other classes such as `Customer`.
- It shows the aggregation relationship between `BankAccount` and `Customer`.
- The diagram also includes methods and constructors for both classes.
public class Customer {
    public Customer(String name, String id) {
        this.name = name;
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public String getID() {
        return id;
    }
    private String name;
    private String id;
}

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, 2844);
        account1.deposit(1000);
        account2.deposit(500);
        account1.transferTo(account3, 100);
        account2.withdraw(300);
        System.out.println("account 1 has "+ account1.getBalance());
        System.out.println("account 2 has "+ account2.getBalance());
    }
}
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());
    }
}