1. 

\[
\begin{array}{c|c|c}
\text{Point A} & \text{sometimes} & \text{sometimes} \\
\text{Point B} & \text{never} & \text{sometimes} \\
\text{Point C} & \text{never} & \text{always} \\
\text{Point D} & \text{sometimes} & \text{sometimes} \\
\text{Point E} & \text{always} & \text{never} \\
\end{array}
\]

2. 

\[
\begin{array}{c|c|c}
\text{Point A} & \text{never} & \text{sometimes} \\
\text{Point B} & \text{sometimes} & \text{always} \\
\text{Point C} & \text{never} & \text{sometimes} \\
\text{Point D} & \text{always} & \text{sometimes} \\
\text{Point E} & \text{sometimes} & \text{never} \\
\end{array}
\]

3. 

```java
public class BankAccount {
    String name;
    double balance;
}
```

4. 

```java
public void deposit(double amount) {
    balance = balance + amount;
}
```

5. 

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

6. 

```java
public class BankAccount {
    String name;
    double balance;
    double transactionFee;

    public void deposit(double amount) {
        balance = balance + amount;
    }

    public void withdraw(double amount) {
        if(balance - amount - transactionFee >= 0) {
            balance = balance - amount;
            balance -= transactionFee;
        }
    }
}
```
public boolean isVertical(Point other) {
    // Note that writing 'this.' is optional. 'x' would also refer to the field.
    // This explicit form comes in handy for clarity in situations involving
    // another instance of the same class.
    return this.x == other.x;
}

public int manhattanDistance(Point other) {
    return Math.abs(this.x - other.x) + Math.abs(this.y - other.y);
}

public double slope(Point other) {
    if(this.x == other.x)
        throw new IllegalArgumentException();
    return ((double)(other.y - this.y)) / (other.x - this.x);
}

public static double euclideanDistance(NDPoint a, NDPoint b) {
    if(a.coordinates.length != b.coordinates.length)
        throw new IllegalArgumentException();
    double differenceSquaredSum = 0;
    for(int i = 0; i < a.coordinates.length; i++) {
        differenceSquaredSum += Math.pow(a.coordinates[i] - b.coordinates[i], 2);
    }
    return Math.sqrt(differenceSquaredSum);
}