And now for something completely different...
A person has a name.

```java
public class Person {
    public final String name;

    public Person(String name) {
        this.name = name;
    }

    // auto-generated hashCode and equals...
}

Person ben = new Person("Ben");
System.out.println(ben.name);
```
A person can be a safety officer.

```java
Person chris = new Person("Chris");
chris.isSafetyOfficer = true;

public class Person {
    public final String name;
    public boolean isSafetyOfficer;

    public Person(String name) {
        this.name = name;
    }

    // auto-generated hashCode and equals...
}
```
A person can have at most one boss.

```java
private Person boss;

public Person(String name, Person boss) {
    this(name);
    this.boss = boss;
}

public Person getBoss() {
    return boss;
}
```

```java
System.out.println(ben.getBoss().name);
System.out.println(maria.getBoss());
```
Does $A$ work with $B$?

$A$ works with $B$ if and only if they have the same boss.

```
public boolean worksWith(Person person) {
    Person myBoss = getBoss();
    Person theirBoss = person.getBoss();
    return myBoss != null &&
           myBoss.equals(theirBoss);
}
```
Is $B$ the employee of $A$?

$B$ is the employee of $A$ if and only if $A$ is $B$'s boss.

```java
public boolean isEmployeeOf(Person person) {
    return person.equals(getBoss());
}
```
Who are A’s employees?

A’s employees are all the people whose boss is A.

```java
public List<Person> getEmployees() {
    ...
}
```
Who are A’s employees?

A’s employees are all the people whose boss is A.

liz.getEmployees()

private List<Person> employees = new LinkedList<Person>();

public Person(String name, Person boss) {
    this(name);
    this.boss = boss;
    boss.addEmployee(this);
}
Who is A’s safety officer?

A’s safety officer is any co-worker of A who is a safety officer.

```
zee.getSafetyOfficer()
ben.getSafetyOfficer()
```

```
public Person getSafetyOfficer() {
    Person myBoss = getBoss();
    if (myBoss == null) {
        return null;
    } else {
        Optional<Person> safetyOfficer =
            myBoss.getEmployees()
                .stream()
                .filter(p -> p.isSafetyOfficer)
                .findFirst();
        if (safetyOfficer.isPresent()) {
            return safetyOfficer.get();
        } else {
            return null;
        }
    }
}
```
Why can’t we just say this?

Maria is Jeff’s boss.
Jeff is Ran’s boss.
Jeff is Liz’s boss.
Ran is Ben’s boss.
Ran is Colleen’s boss.
Liz is Zee’s boss.
Liz is Chris’s boss.

Chris is a safety officer.

Person A is an employee of person B if B is A's boss.

Person A works with person B if they have the same boss.

Person A’s safety officer is anyone A works with who is a safety officer.
Behold: Prolog!

boss(maria, jeff).
boss(jeff, ran).
boss(jeff, liz).
boss(ran, ben).
boss(ran, colleen).
boss(liz, chris).
boss(liz, zee).
safetyOfficer(chris).

employee(PersonA, PersonB) :- boss(PersonB, PersonA).

worksWith(PersonA, PersonB) :- boss(Boss, PersonA), boss(Boss, PersonB).
safetyOfficer(PersonA, PersonB) :- sameDivision(PersonA, PersonB),
                                 safetyOfficer(PersonA).