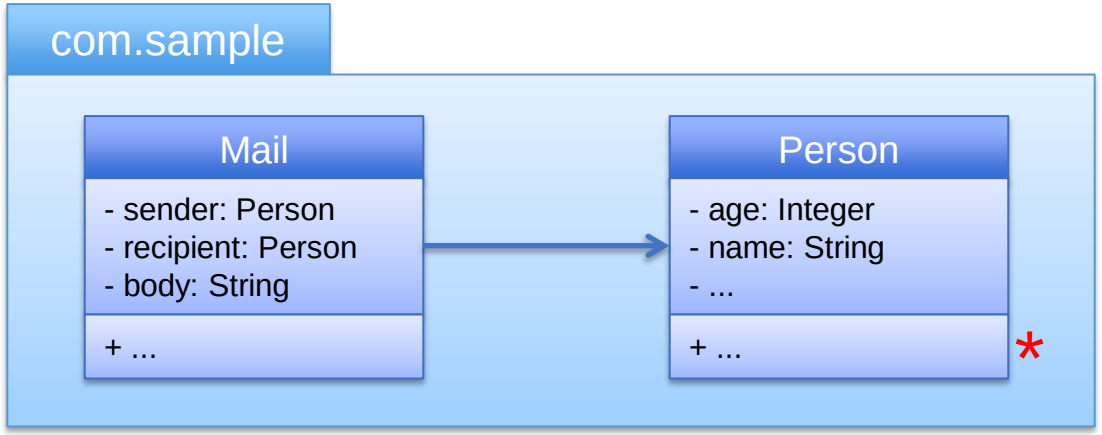
Drools exercises

Lorenzo Rossi

LCP

- Create new Drools project

- Create the classes Main and Person



- Create a new rule file
- Populate the WM with some person

```
Person p1 = new Person("Bob", 35);
Person p2 = new Person("Molly", 22);
Person p3 = new Person("Bob", 8);
Person p4 = new Person("Anna", 15);
Person p5 = new Person("Bob", 25);
Person p6 = new Person("Sandra", 40);
ksession.insert(p1);
ksession.insert(p2);
ksession.insert(p3);
ksession.insert(p4);
ksession.insert(p5);
ksession.insert(p6);
ksession.insert(new Mail(p1, p4,
  "Do your homeworks!"));
ksession.insert(new Mail(p5, p5,
  "Remember to do the homeworks!"));
...
ksession.fireAllRules();
```

- Write a rule that prints out every mail content

- Write a rule that prints out every person information

- Write a rule that prints out every mail content
- Write a rule that prints out every person information

```
package com.sample

import com.sample.Mail;
import com.sample.Person;

rule "Rule 1"
when
  $m: Mail()
then
  System.out.println("r1-Mail: " + $m);
end

rule "Rule 2"
when
  $p: Person()
then
  System.out.println("r2-Person: " + $p);
end
```

- Finds all person named "Bob" with less than ten years old, or between the ages of 18 and 35 years old

- Finds all person named "Bob" with less than ten years old, or between the ages of 18 and 35 years old

```
package com.sample

import com.sample.Mail;
import com.sample.Person;

rule "Rule 3"
when
  $p: Person( name == "Bob",
        age < 10 ||
        (age >= 18 && age <= 35)
  )
then
  System.out.println("r3-Person: " + $p);
end
```

- Write a rule that finds all the couples of people

- Write a rule that finds all the couples of people

```
rule "Rule 4"
when
  $p1: Person()
  $p2: Person()
then
  System.out.println("r4: " + $p1 +
" vs. " + $p2);
end
```

- Write a rule that finds all the couples of people

- Write a rule that finds all the couples of people avoiding to couple people with themselves

- Write a rule that finds all the couples of people

```
rule "Rule 4"
when
  $p1: Person()
  $p2: Person()
then
  System.out.println("r4: " + $p1 +
" vs. " + $p2);
end
```

- Write a rule that finds all the couples of people avoiding to couple people with themselves

```
rule "Rule 5"
when
  $p1: Person()
  $p2: Person( this != $p1 )
then
  System.out.println("r5: " + $p1 +
" vs. " + $p2);
end
```

- Write a rule that finds all the couples of homonym people avoiding to couple people with themselves

- Write a rule that finds all the couples of homonym people avoiding to couple people with themselves

```
rule "Rule 6"
when
  $p1: Person( $n: name )
  $p2: Person( this != $p1,
               name == $n )
then
  System.out.println("r6: " + $p1 +
" vs. " + $p2);
end
```

- Write a rule that finds all the couples of homonym people avoiding to couple people with themselves and to repeat reverse coupling (p1,p2 but not p2,p1)

- Write a rule that finds all the couples of homonym people avoiding to couple people with themselves and to repeat reverse coupling (p1,p2 but not p2,p1)

```
rule "Rule 7"
when
  $p1: Person( $n: name )
  $p2: Person( this != $p1, $p1 < $p2,
               name == $n )
then
  System.out.println("r7: " + $p1 +
" vs. " + $p2);
end
```

- Write a rule printing the content of mails sent by person named "Bob", that are at least 20, to himself

- Write a rule printing the content of mails sent by person named "Bob", that are at least 20, to himself

```
rule "Rule 8"
when
  $p:  Person( age >= 20,
         name == "Bob" )
  $pp: Person( this == $p )
  $m:  Mail ( sender == $p,
         recipient == $pp,
         $b: body )
then
  System.out.println("r8: " + $b);
end
```

- Write a rule printing the content of mails sent by person named "Bob", that are at least 20, to himself

- Write a rule printing the content of mails sent by person named "Bob", that are at least 20, to himself

```
rule "Rule 9"
when
 Mail ( $p: sender,
         sender.name == "Bob",
         sender.age >= 20,
         recipient == $p,
         $t: body.toString() )
then
  System.out.println("r9: " + $t);
end
```

Drools supports advanced operators like ∃ and ∀

- `exists P(…)`
  the WM contains at least one fact matching
- `not P(…)`
  the WM does not contain any matching fact
- `forall P(…)`
  all the objects of type P in the WM match

- Write a rule printing person
  - Who received at least one mail

```
rule "Rule 10"
when
  $p: Person( $n: name )
  exists Mail( recipient == $p )
then
  System.out.println("r10: " + $n
    + " received a mail");
end
```

- Write a rule printing person
  - Who received at least one mail
  - Who had not received any mail

```
rule "Rule 10"
when
  $p: Person( $n: name )
  exists Mail( receiver == $p )
then
  System.out.println("r10: " + $n
    + " received a mail");
end

rule "Rule 11"
when
  $p: Person( $n: name )
  not Mail( recipient == $p )
then
 System.out.println("r11: " + $n
    + " did not received mail");
end
```

- Write a rule printing person
  - Who received at least one mail
  - Who had not received any mail
  - Who received mail only from Bob

```
rule "Rule 12"
when
  $p: Person( $n: name )
  forall (
    Mail( $s: sender, recipient == $p )
    Person( this == $s, name == "Bob" )
  )
then
  System.out.println("r12: " + $n
    + " received mail only from Bob");
end
```

- Keyword **from** permits to access object in a Collection

```
rule "Example of FROM"
when
  $l: List()
  $p: Person() from $l
then
  System.out.println($l + "/" + $p);
end
```

- Keyword **`collect`** is dual to **`form`** it produces a collection of facts on the basis of the selection

```
rule "Example of COLLECT"
when
  $c: collect ( Person() )
  $p: Person() from $c
then
  System.out.println($c + "/" + $p);
end
```

- Keywork **accumulate** works similarly to **collect**
- It performs operation on collection of facts
- Exist specific operators for: **min**, **max**, **count**, ecc.

```
rule "Esempio di Funzioni notevoli"
when
  accumulate( Person( $a: age ),
    $max: max( $a ),
    $min: min( $a ),
    $avg: average( $a )
  )
then
  System.out.println(
      "M:" + $max + " m:" + $min + " a:" + $avg);
end
```

- Drools permits to query the WM without affecting it
- Query are used tofilter facts without applying any action

```
query "Query: Q1"
  $p: Person( age > 18 )
end

query "Query: Q2" ( int $a )
  Person( $n: name, age == $a )
end

query "Query: Q3" ( String $n, int $a )
  Person( $n := name, $a := age )
end
```