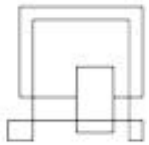
Life is great
VIA University College



Software Development with UML and Java 2

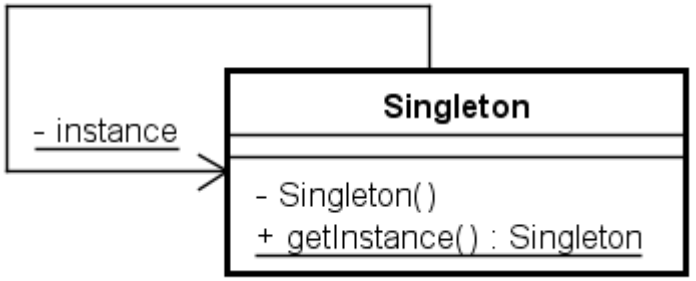
Agenda

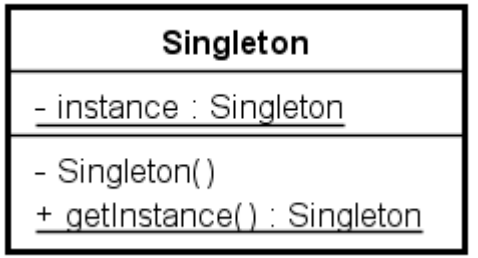
- Design pattern: Singleton
- Design pattern: Flyweight

Design Pattern: Singleton

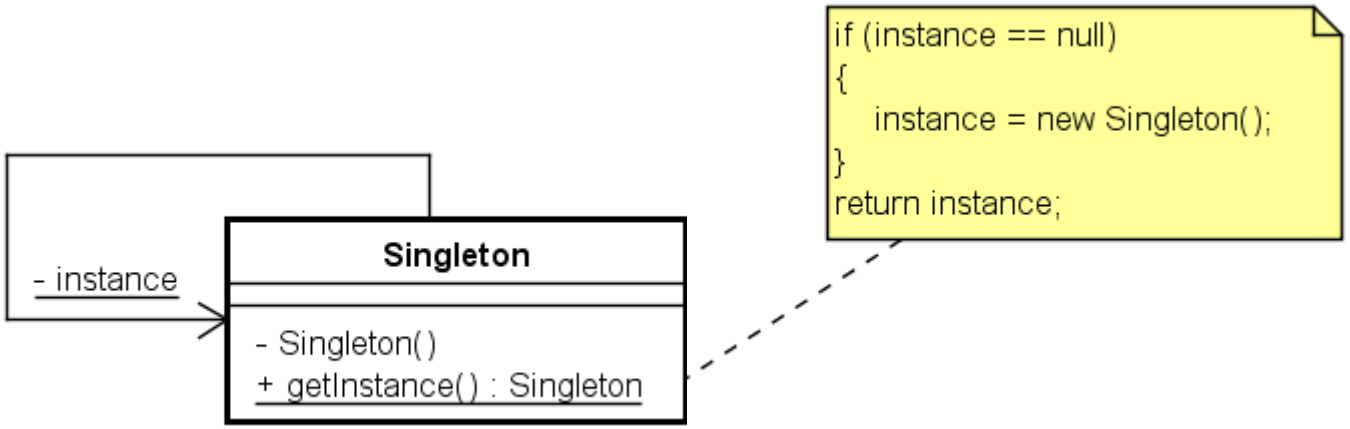
- Intent
  - Ensure that a class only has one instance, and provide a global point to access it
- Example
  - You have a network with many printers
  - All printers must be served by one single (global) print spooler
- Problem
  - How to create a single print spooler class that every printer driver has access to global spooler?

Design Pattern: Singleton





Design Pattern: Singleton



- Why is the Constructor private?
- Why is method getInstance() static?

Singleton implementation

```
public class Singleton
{
  private static Singleton instance;

  private Singleton()
  {
    // ...;
  }
  public static Singleton getInstance()
  {
    if (instance == null)
    {
      instance = new Singleton();
    }
    return instance;
  }
  // ...
}
```

Private class variable of the same type as the class (class variable = static)

Private constructor

Public class method returning an instance of the class (class method = static)

Note: this method has to ensure that only one instance is created

Singleton use

```
public class SingletonTest
{
  public static void main(String[] args)
  {
    Singleton reference1 = Singleton.getInstance();
    Singleton reference2 = Singleton.getInstance();

    System.out.println("reference1=" + reference1);
    System.out.println("reference2=" + reference2);
  }
}

/* OUTPUT:
reference1=Singleton@19821f
reference2=Singleton@19821f
*/
```

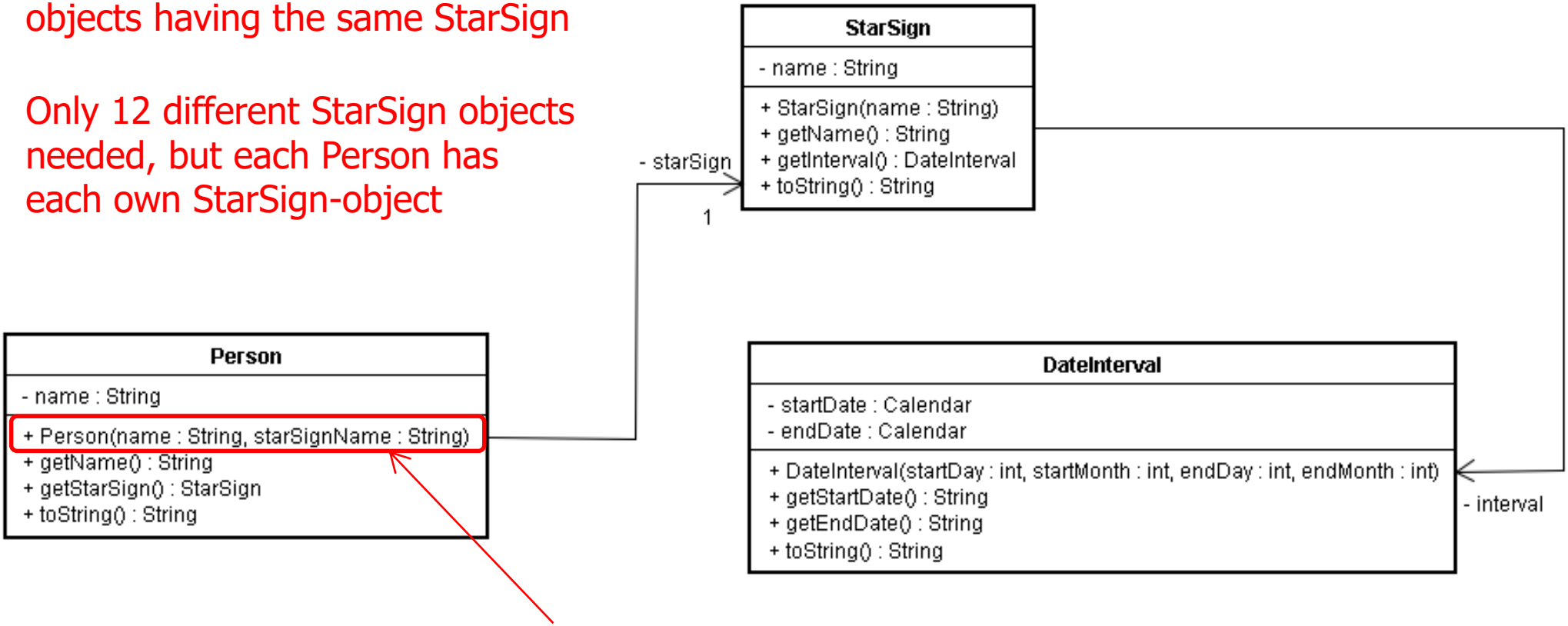
Design Pattern: Flyweight

- Intent
  - To share objects instead of creating more objects with the same state
- Example
  - Person list with information of star sign. The same star sign is shared by more than one person
  - A Person with a nationality. The same nationality is shared by more than one person
  - A calendar with each task having a Date object. The same date is shared by a lot of calendar tasks
- Problem
  - How to use a factory (a pool) of objects?

Motivation for Flyweight: A Person class

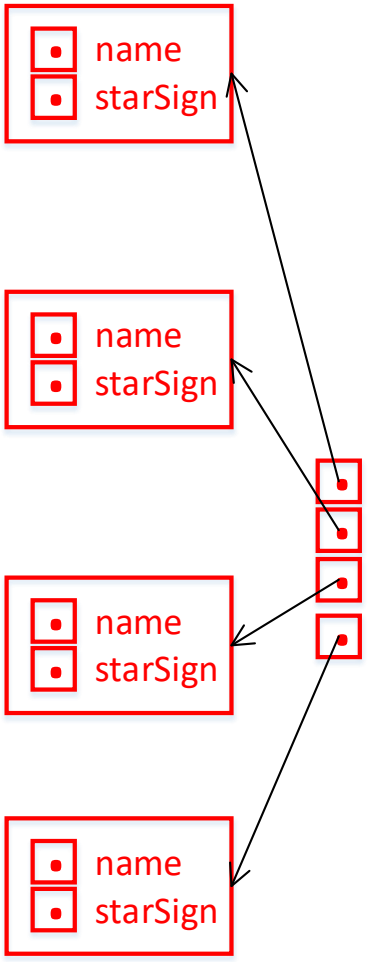
There could be multiple Person objects having the same StarSign

Only 12 different StarSign objects needed, but each Person has each own StarSign-object

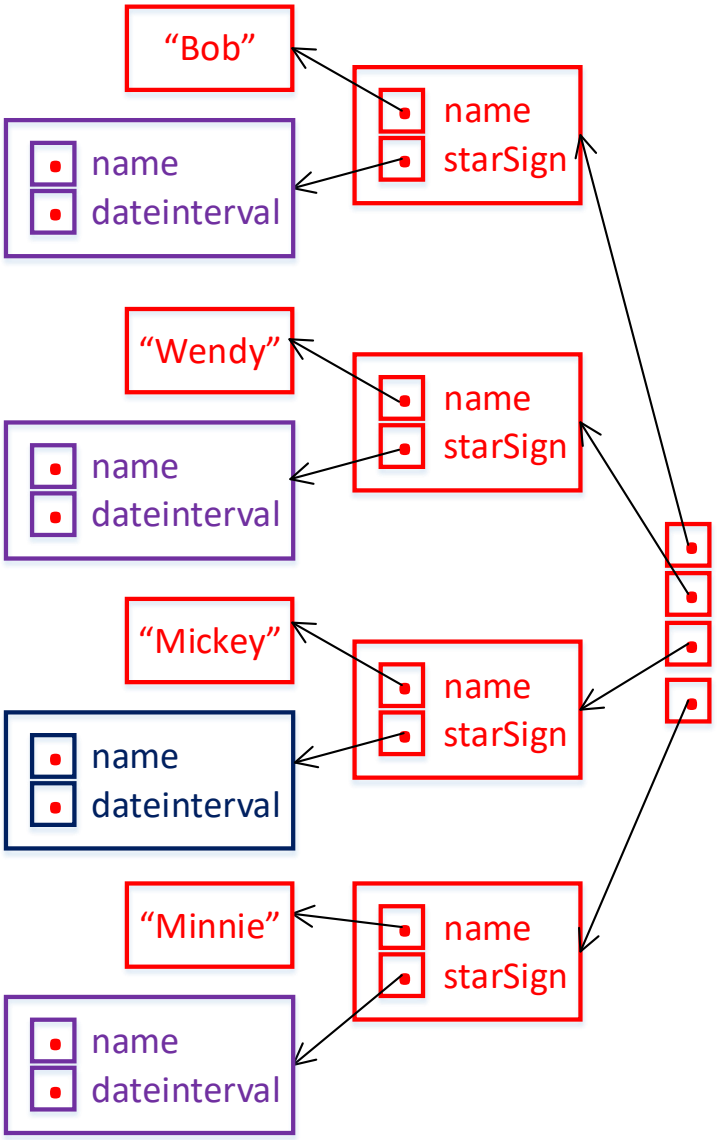



```
public Person(String name, String starSignName)
{
   this.name = name;
   this.starSign = new StarSign(starSignName);
}
```

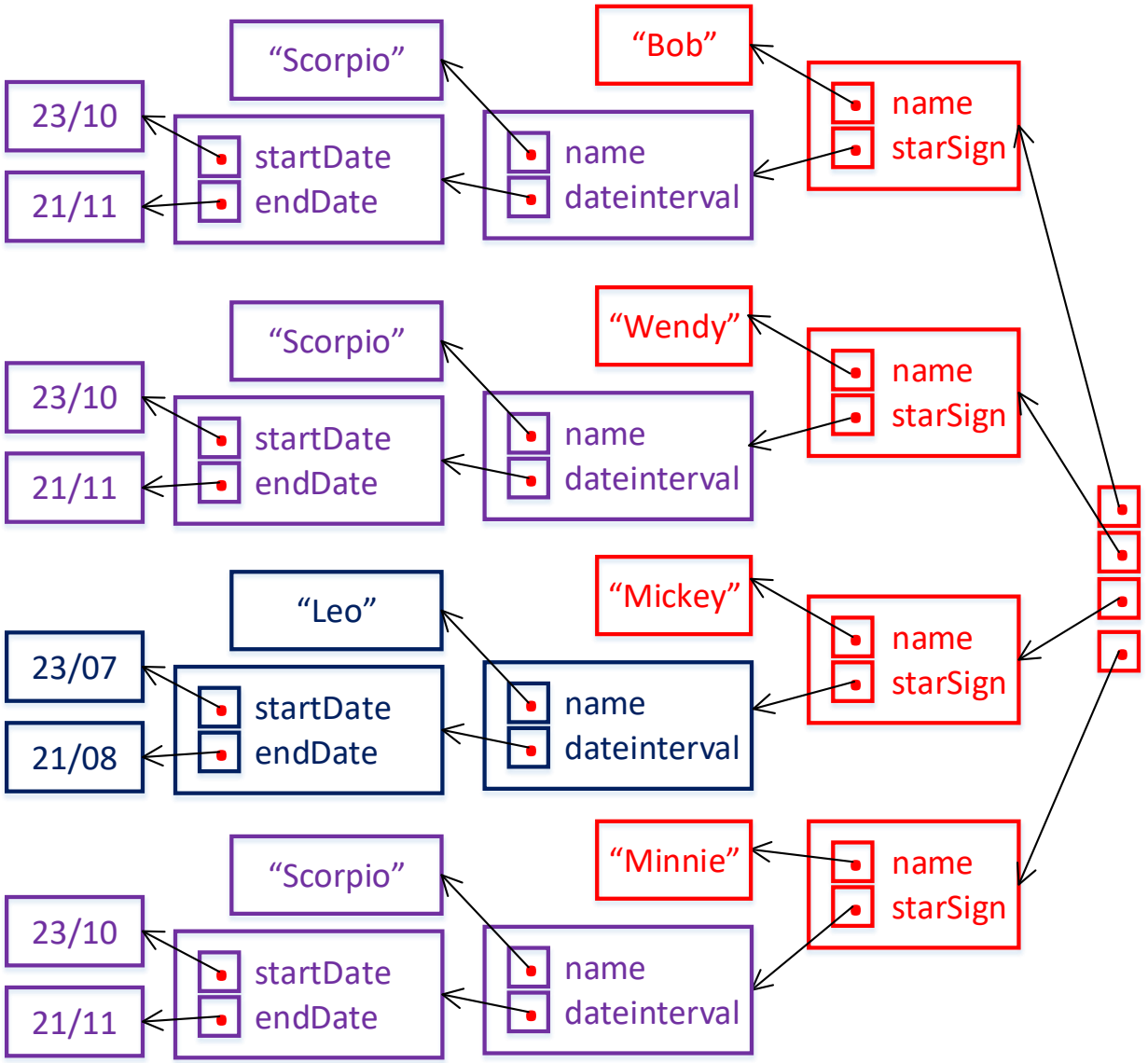
Example: 4 person objects



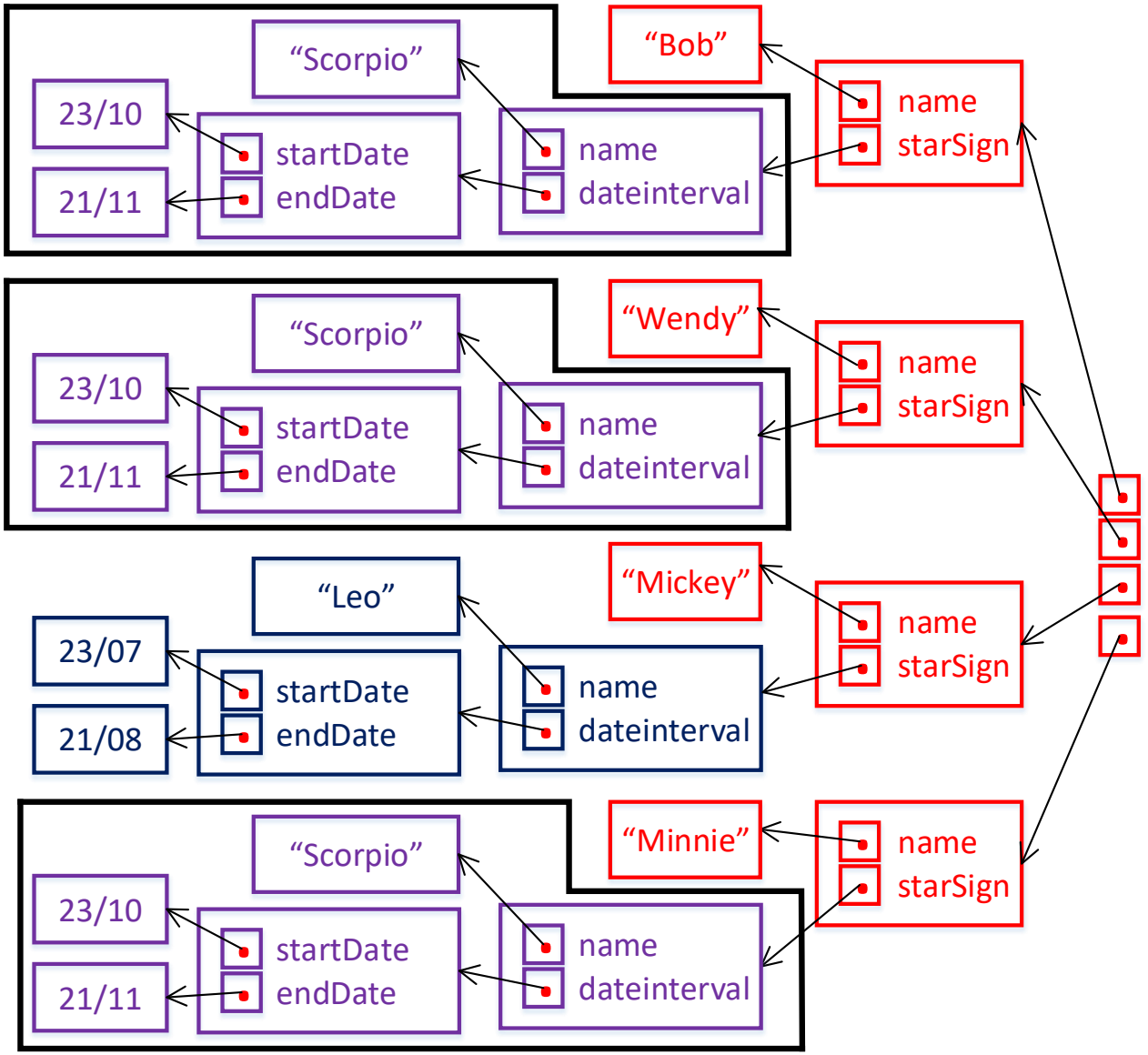
Example: 4 person objects



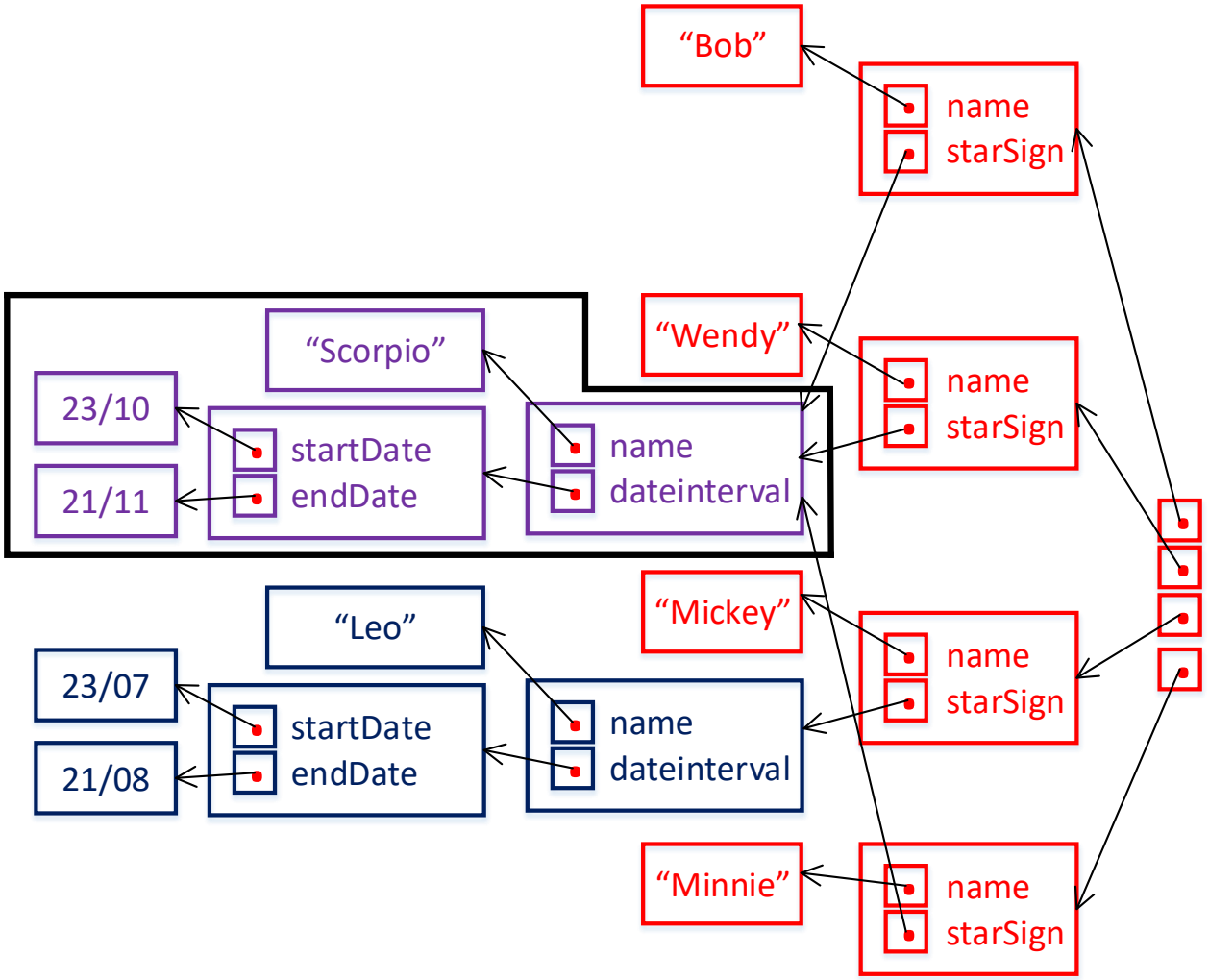
Example: 4 person objects



Example: 4 person objects

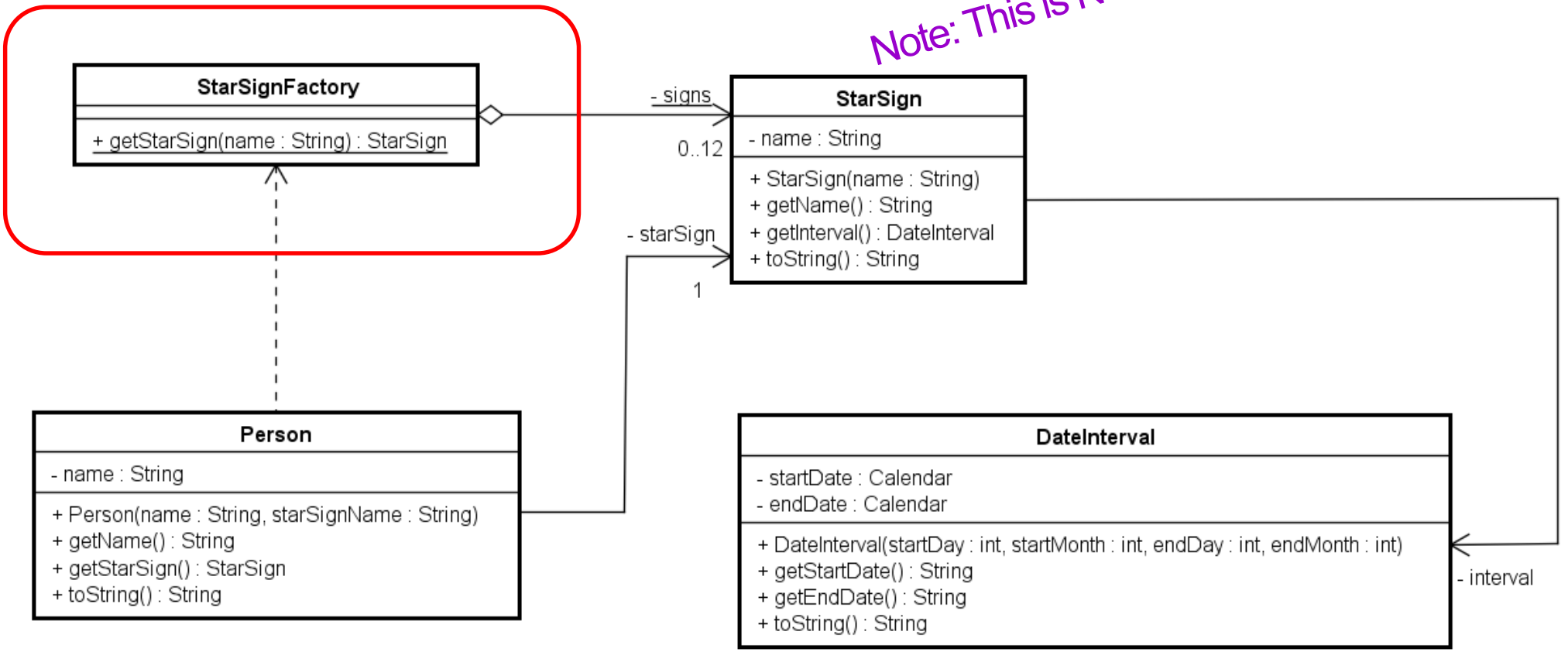


Example: 4 person objects (sharing objects)



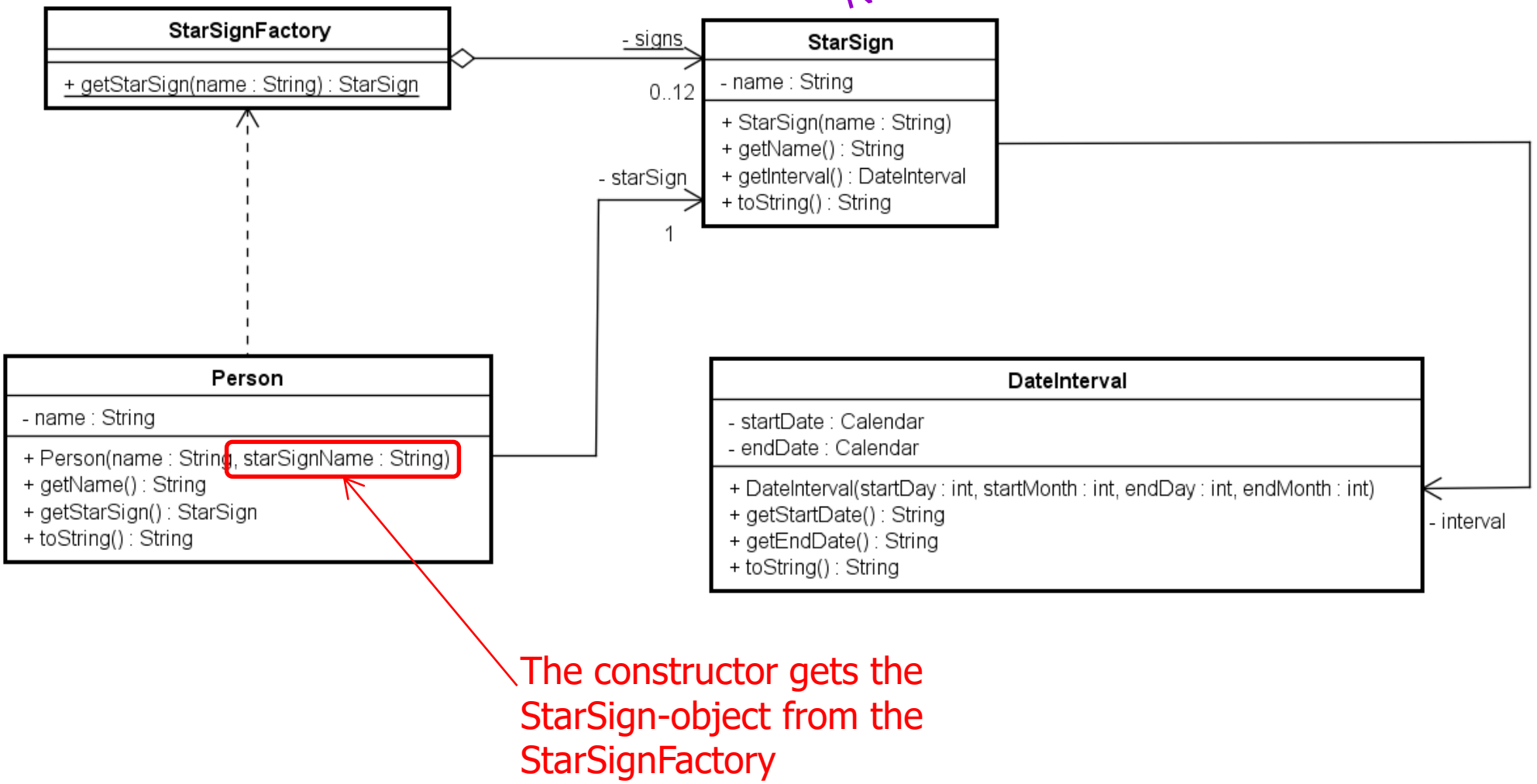
A factory producing StarSign objects

Note: This is NOT a Flyweight DP - yet



A factory producing StarSign objects

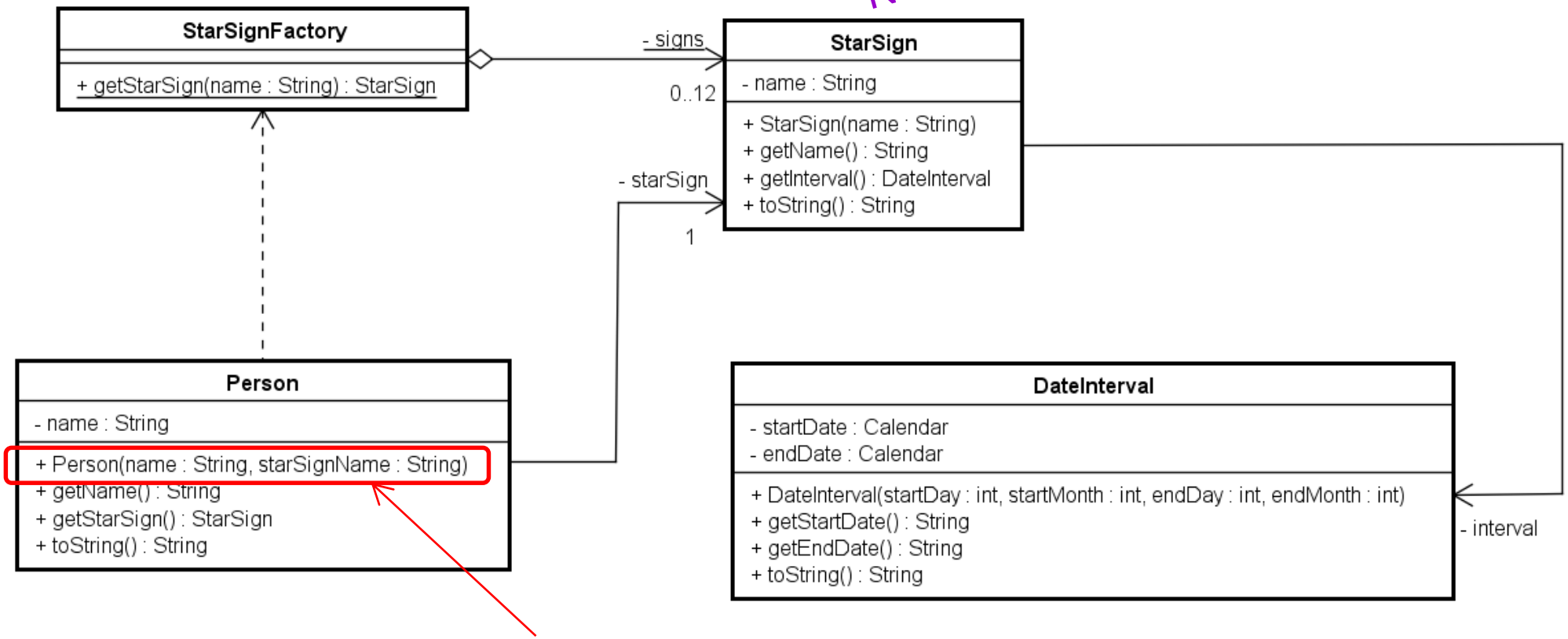
Note: This is NOT a Flyweight DP - yet



The constructor gets the StarSign-object from the StarSignFactory

A factory producing StarSign objects

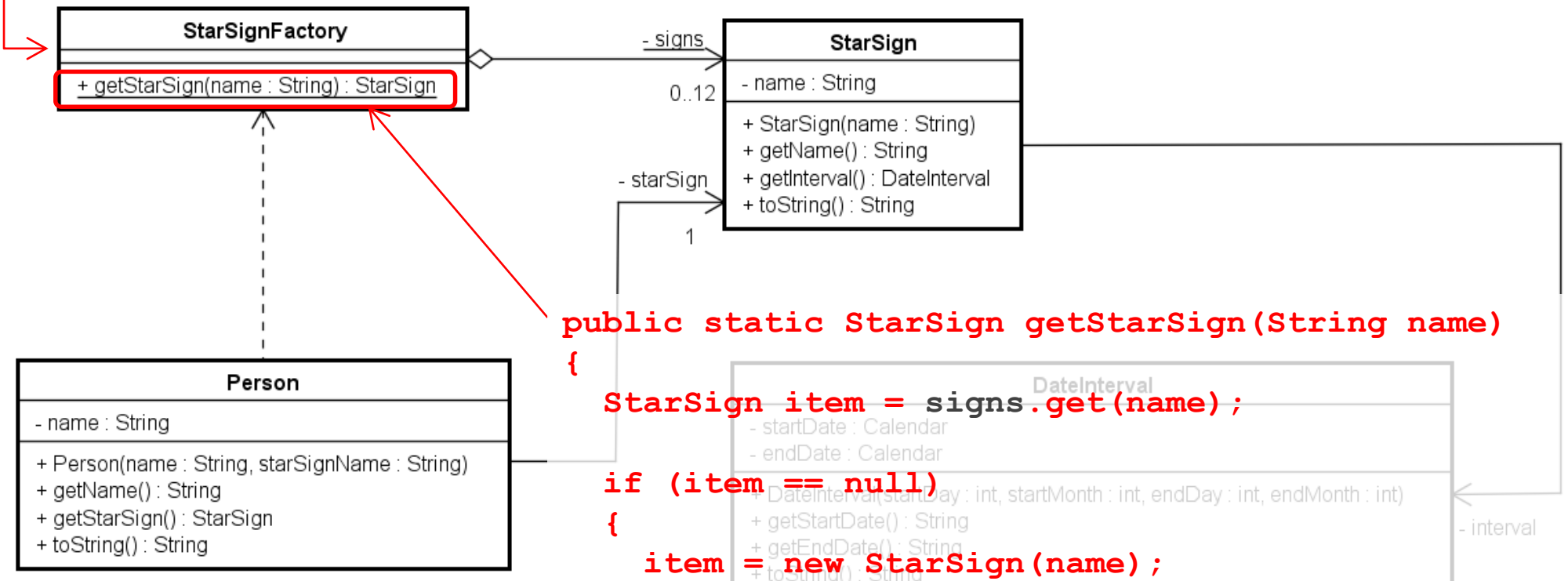
Note: This is NOT a Flyweight DP - yet




```
public Person(String name, String starSignName)
{
   this.name = name;
   this.starSign = StarSignFactory.getStarSign(starSignName);
}
```

A factory producing StarSign objects

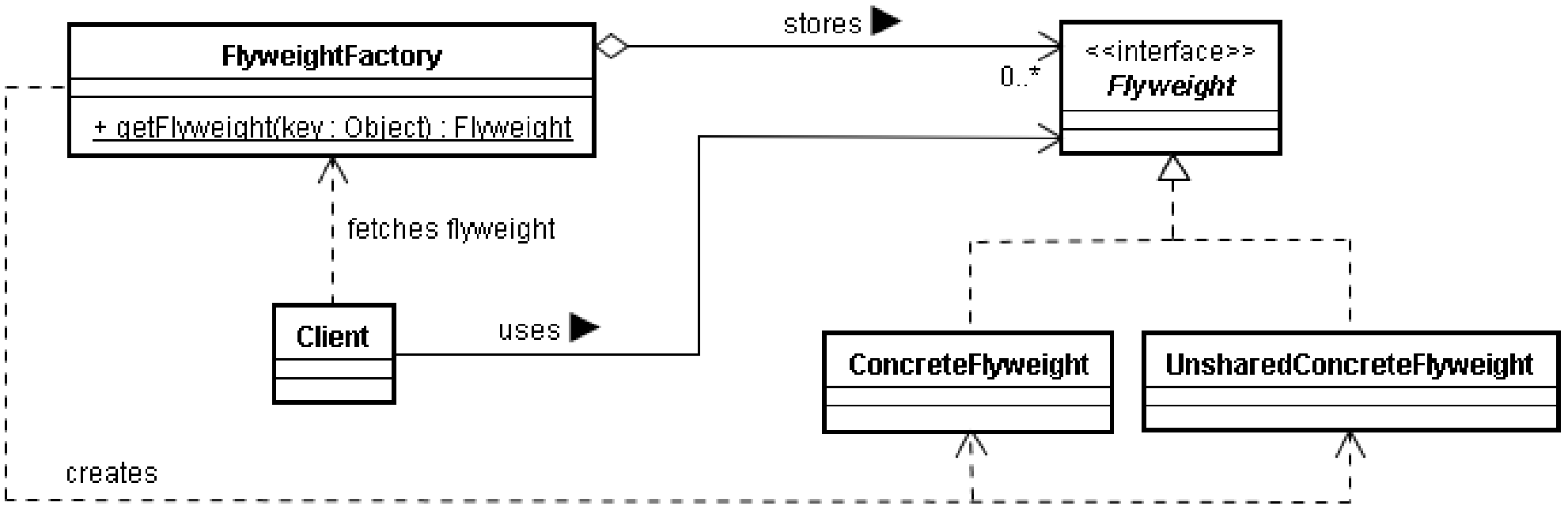
```
private static HashMap<String, StarSign> signs = new HashMap<String, StarSign>();
```



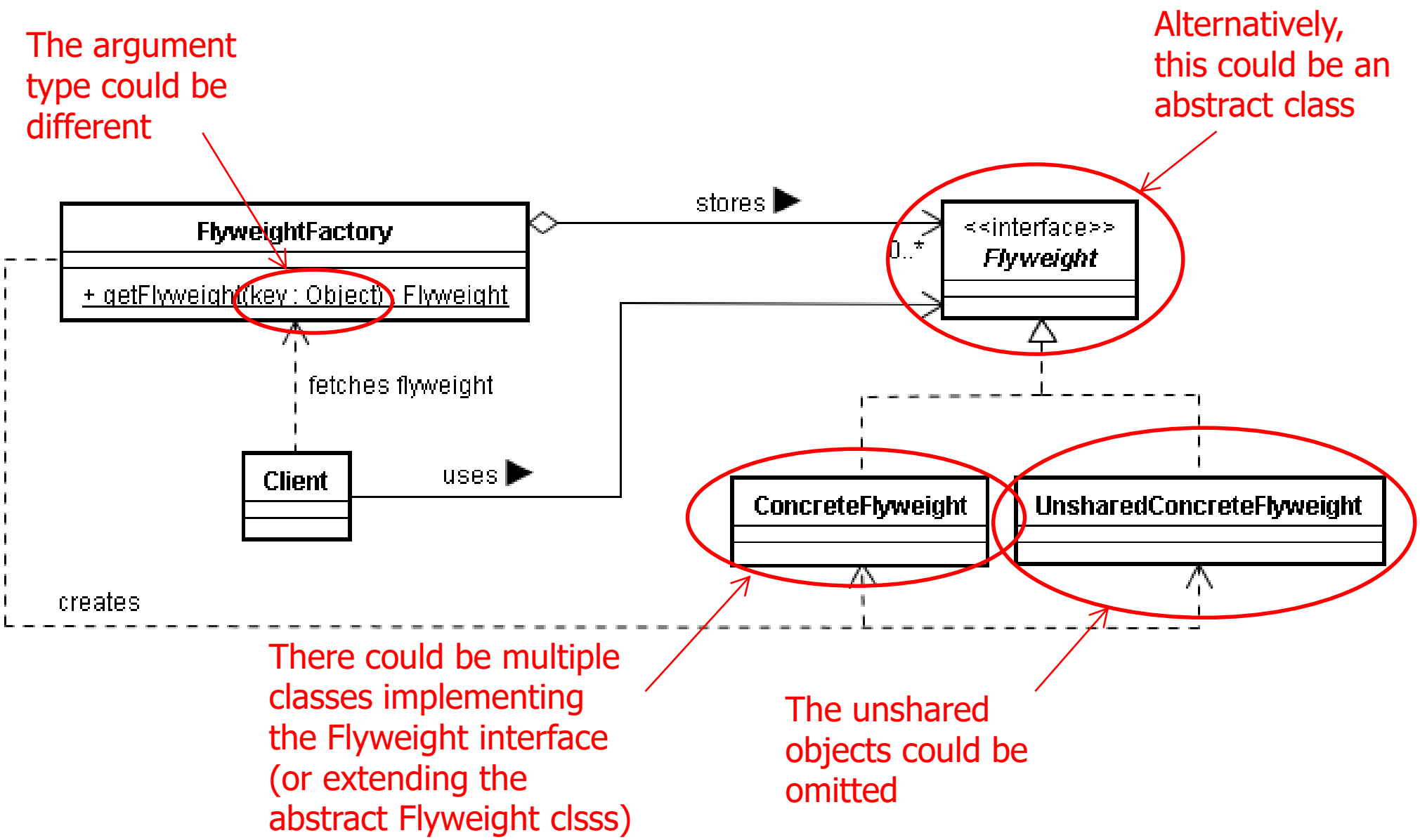

```
public static StarSign getStarSign(String name)
{
  StarSign item = signs.get(name);

  if (item == null)
  {
    item = new StarSign(name);
    signs.put(name, item);
  }
  return item;
}
```

Note: This is NOT a Flyweight DP - yet

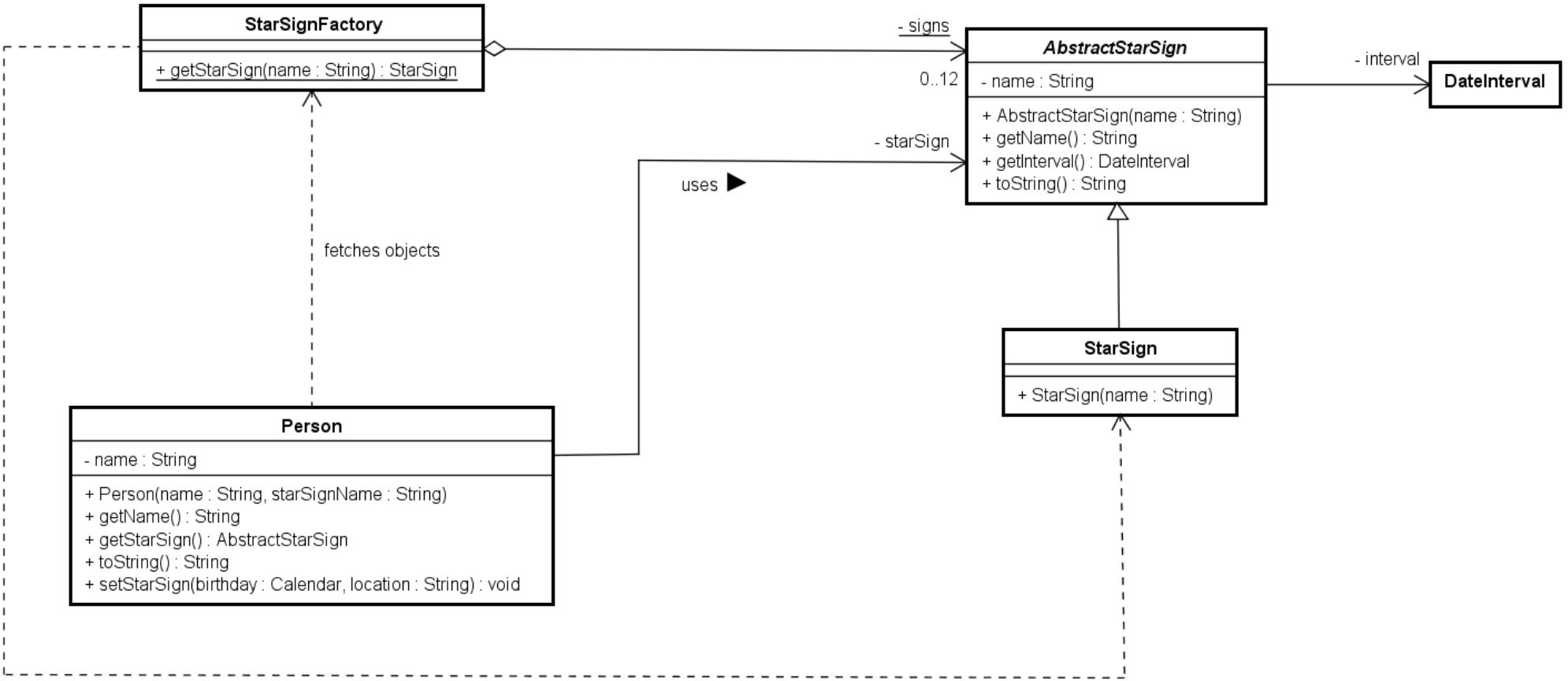
Flyweight design pattern



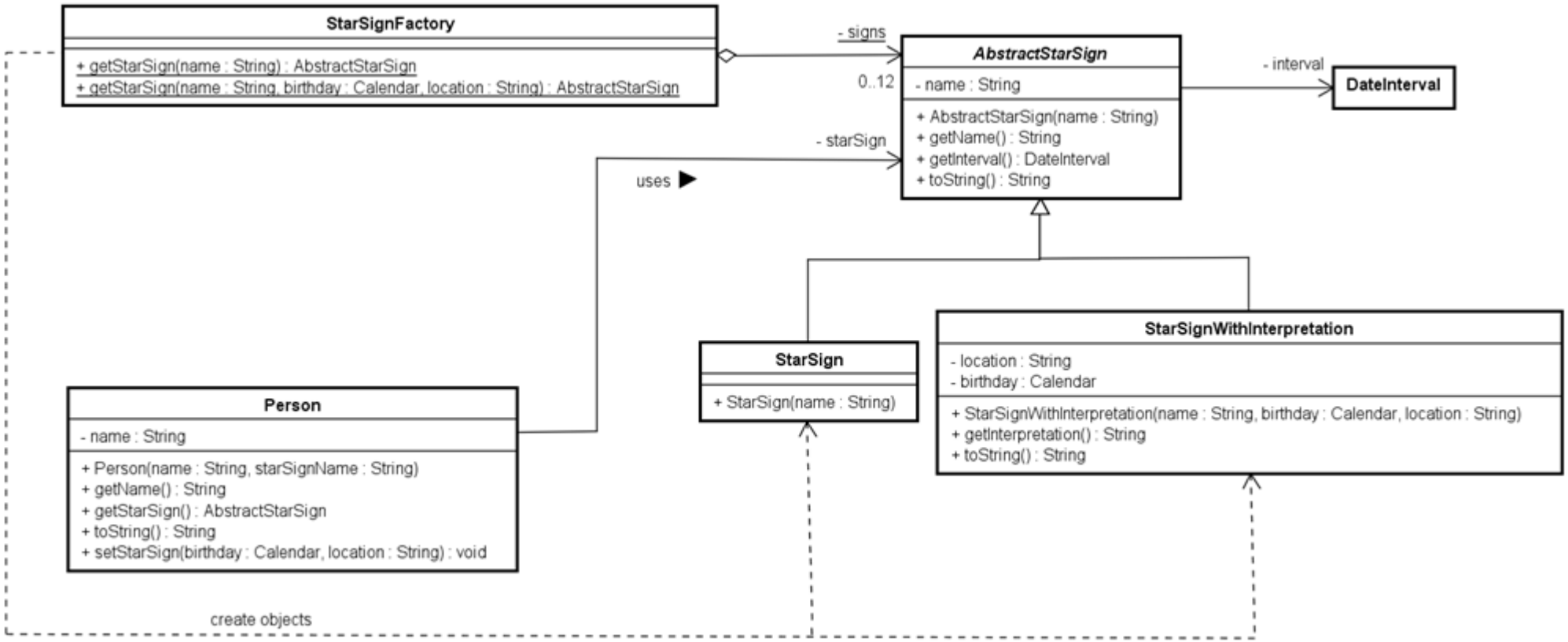
Flyweight design pattern



StarSign – (Flyweight – only shared objects)



StarSign – shared and unshared obj (Flyweight)



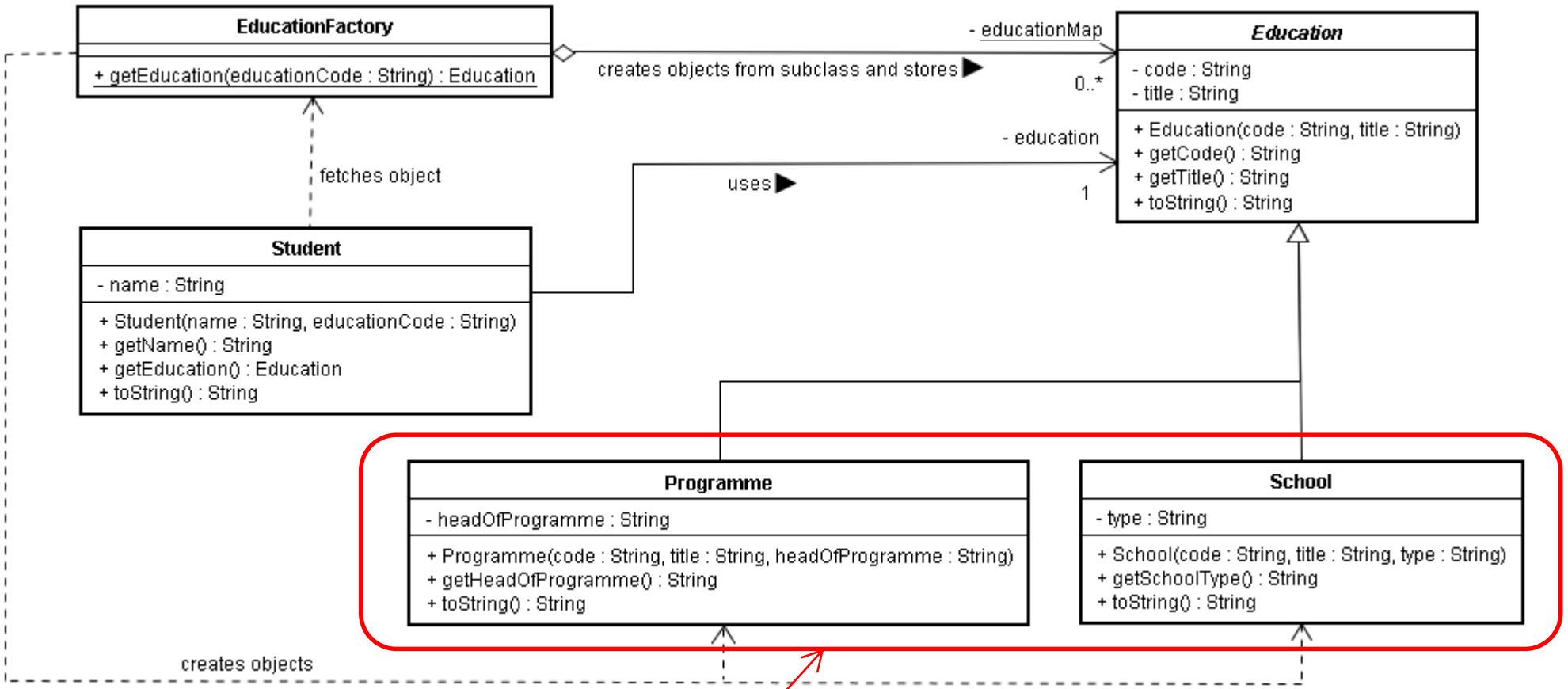
Another Flyweight Example: Education

- A Student has a name and an Education
  - Education is either a Programme or a School
- A lot of students share the same Education

- Example 1:
  - ”Wendy” is at a Programme
    - Code = ”ICT”
    - Title = ”ICT Engineering”
    - headOfProgramme = ”Sam”

- Example 2:
  - ”Bob” is at a School
    - Code = ”HJS”
    - Title = ”Horsens Junior School”
    - Type = ”Elementary School”

Education – (Flyweight – only shared objects)



Two ”shared-objects”
subclasses of the
abstract Flyweight (Education)

Student (Flyweight)