(11224) INTRODUCTION TO SOFTWARE ENGINEERING

Lecture 18: Code Engineering

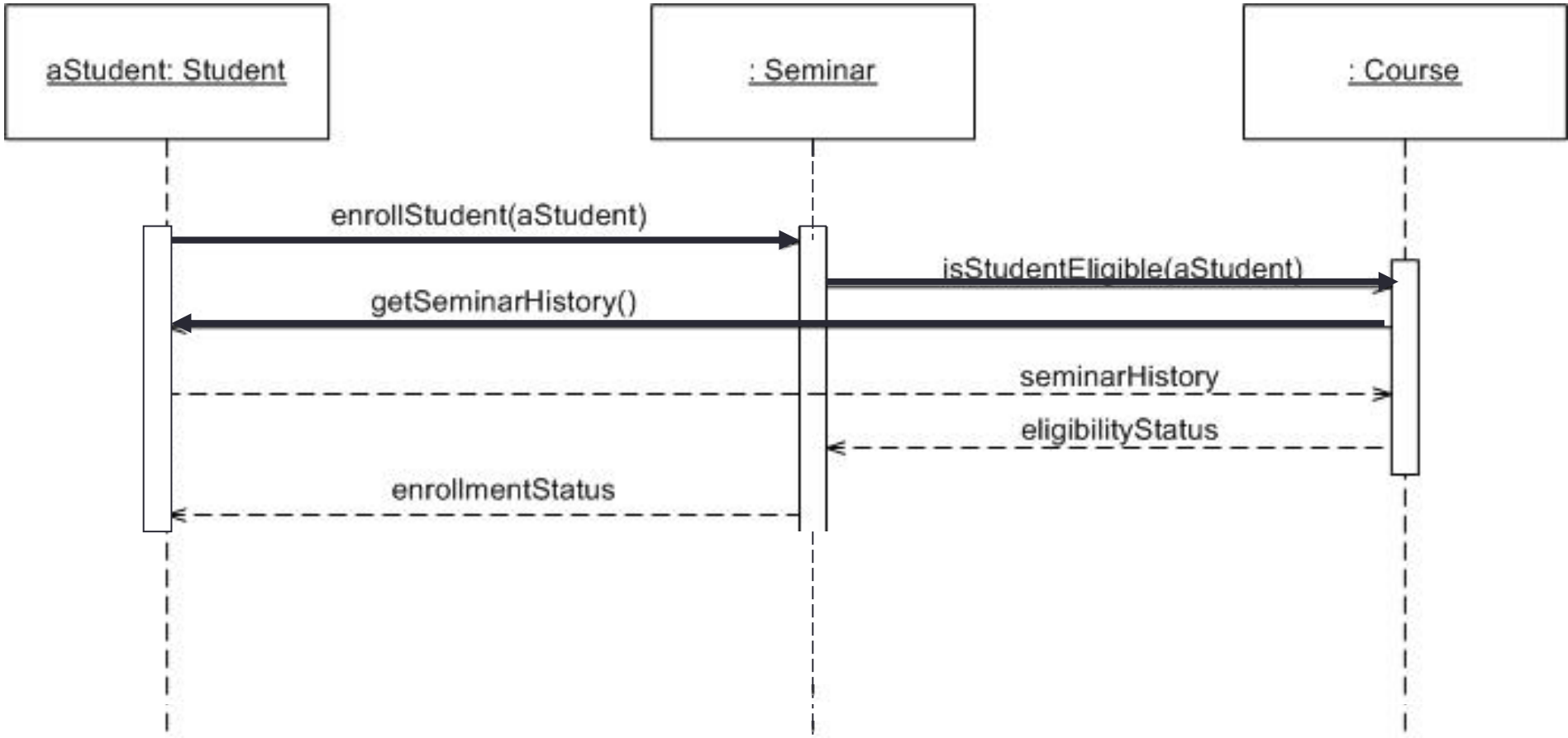
How to implement a JAVA code from a sequence diagram

Shereen Fouad

Announcements

- Online test on Friday the ---- of March at 10 am and will close (same day) at 10 pm
  - Counts for 5% of the entire Module Mark
  - Test will be Multiple Choice Questions (25 questions)
  - Once you begin the online test you have only 60 minutes to complete it.
  - You only have one attempt to complete the online test.
  - It will cover all concepts discussed in this module.
- Past years exams are available on canvas.

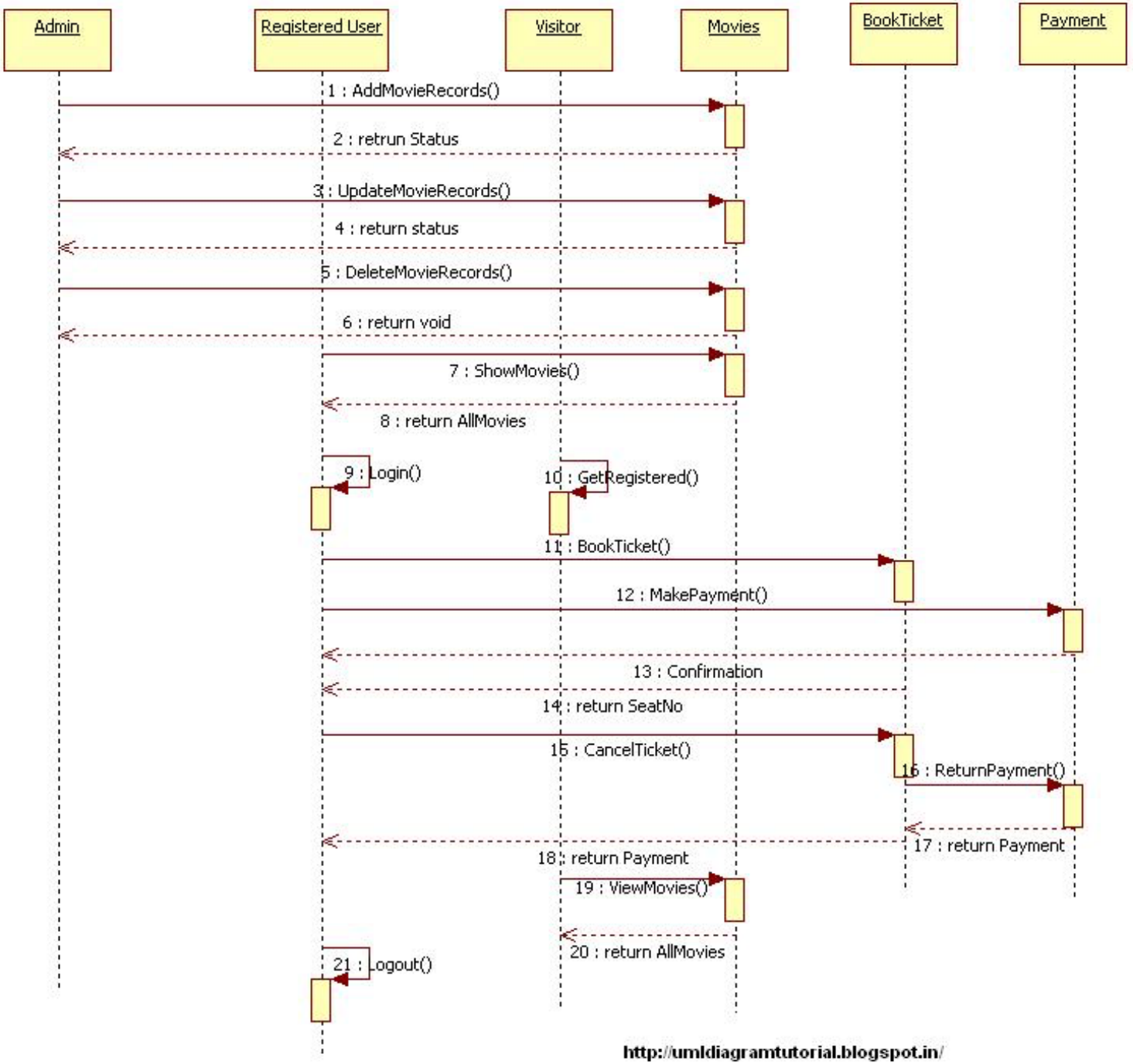
Sequence Diagram



Exercise

- Draw a sequence diagram for booking a movie ticket online

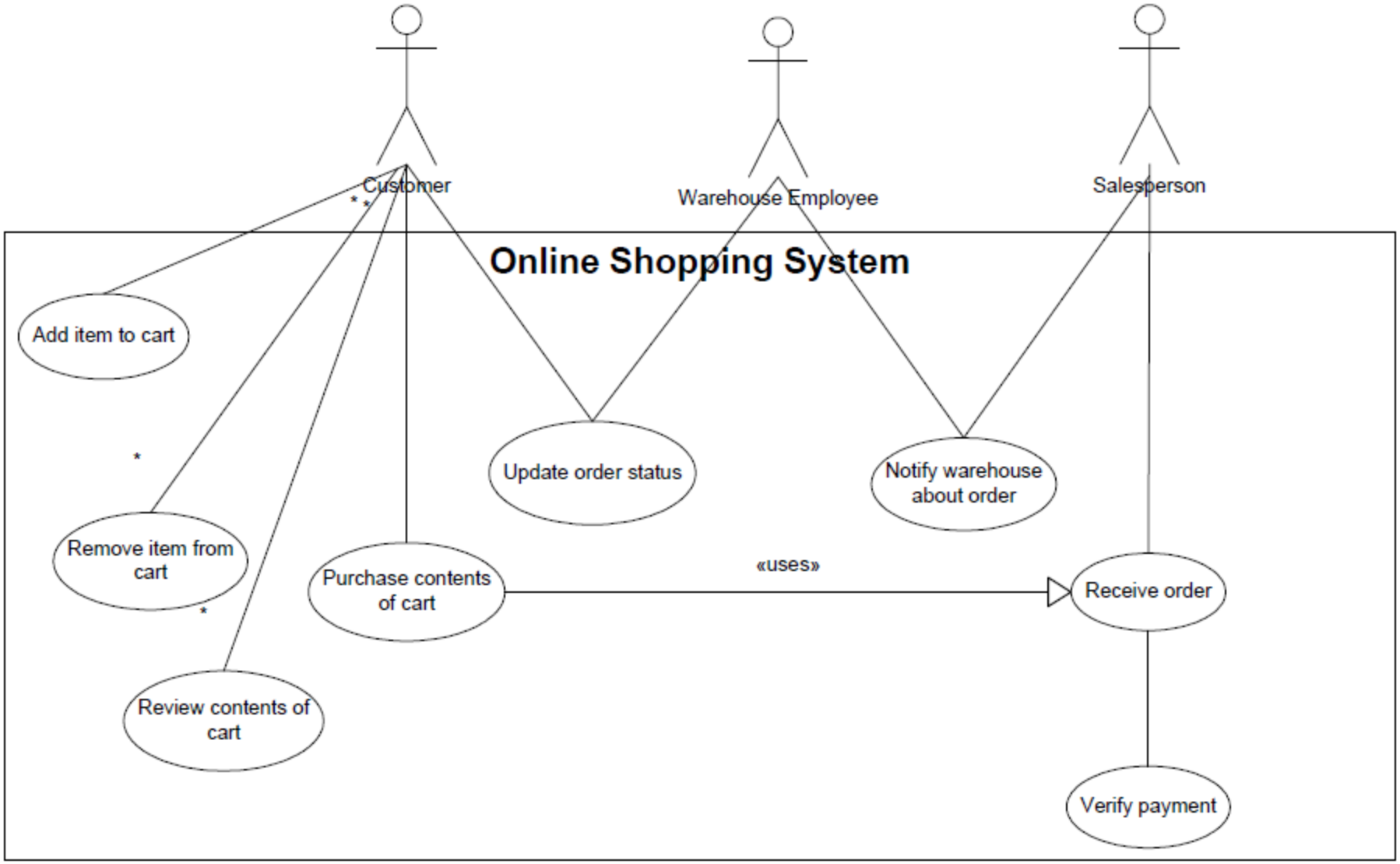
- Treat the following parts as a separate class
    - Admin
    - Registered user
    - Unregistered user
    - Movies
    - Book ticket
    - Payment



http://umldiagramtutorial.blogspot.in/

Exercise

Develop a sequence diagram showing the interactions involved in the below use case



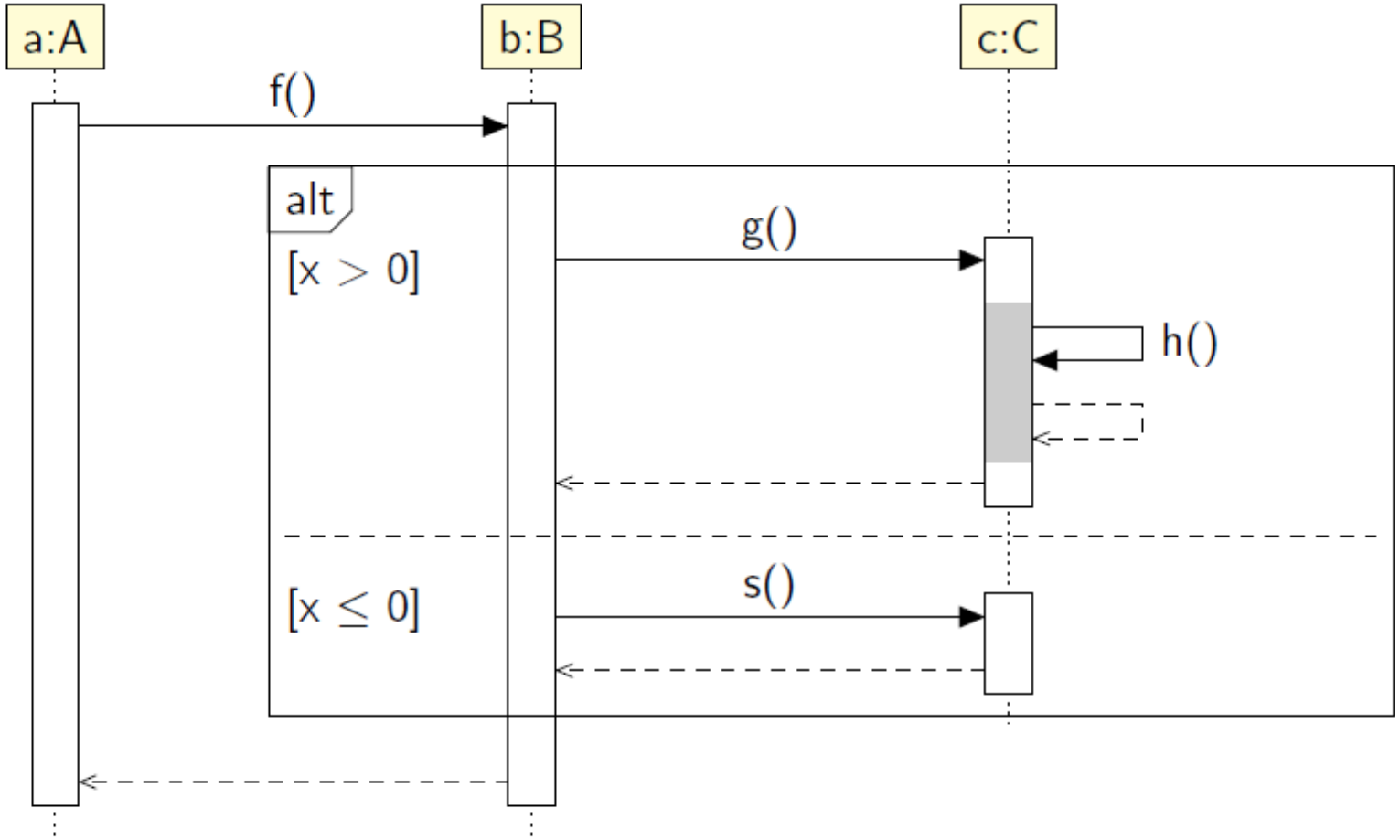
Reverse-engineer Source Code into UML Diagrams

- **The ObjectAid UML Explorer for Eclipse**
- http://www.objectaid.com/ (free)
- **Visual Paradigm**
- http://www.visual-paradigm.com/
- Round-trip engineering helps keep your Java source code and software design synchronized.

Code Engineering

- Sequence diagrams can be somewhat close to the code level.
- It is hard to generate a full correct running code automatically from the sequence diagram.
- So why not just code it?

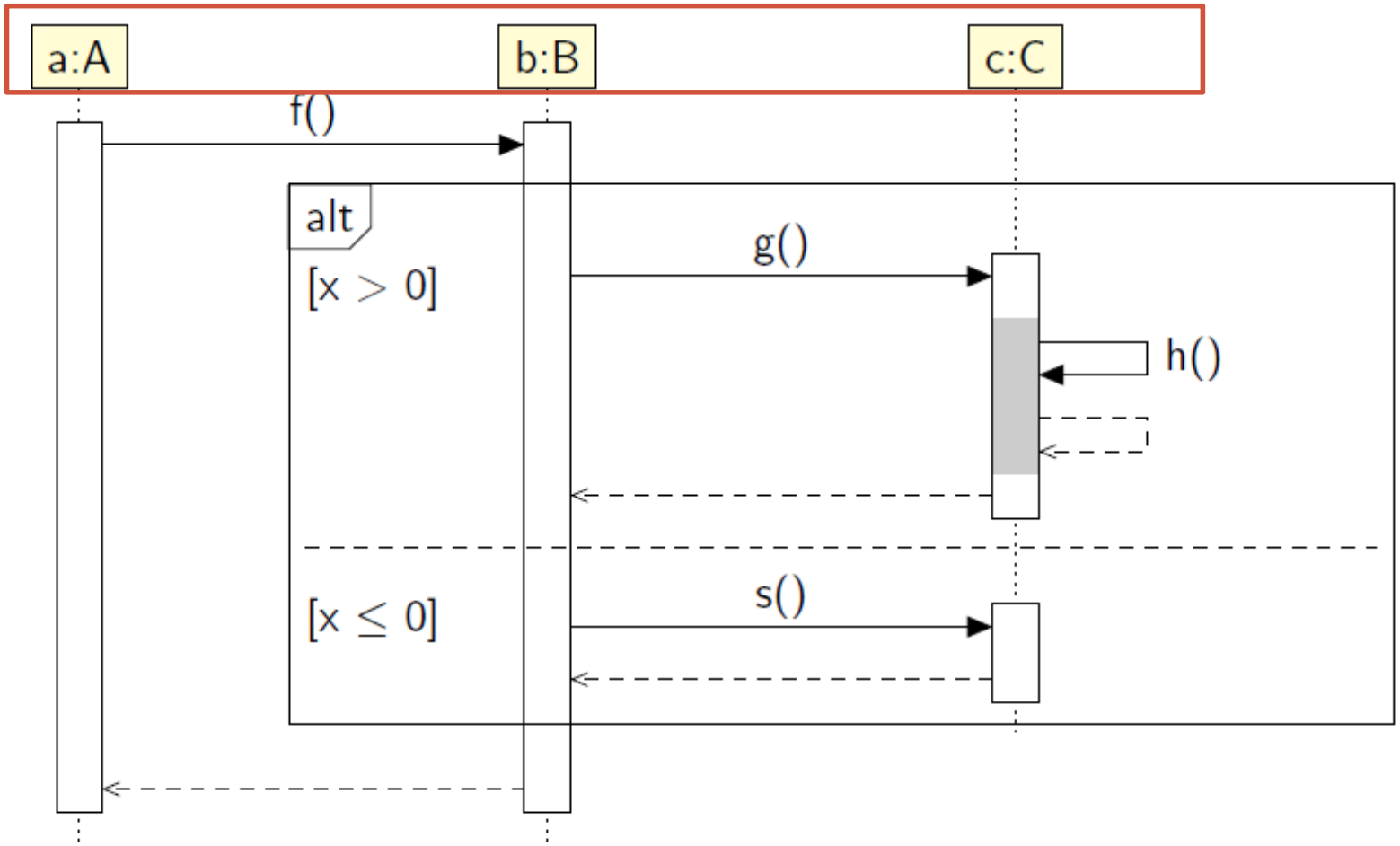
Write the minimal Java code for the classes `A`, `B` and `C` that captures all the behaviour in the following UML Sequence Diagram. The bodies of all methods other than `f()`, `g()`, `h()` and `s()` should be left empty. Constructors and properties (i.e. get and set methods) for the classes do not need to be shown.



Rules of Thumb to translate the Sequence Diagram into classes and methods

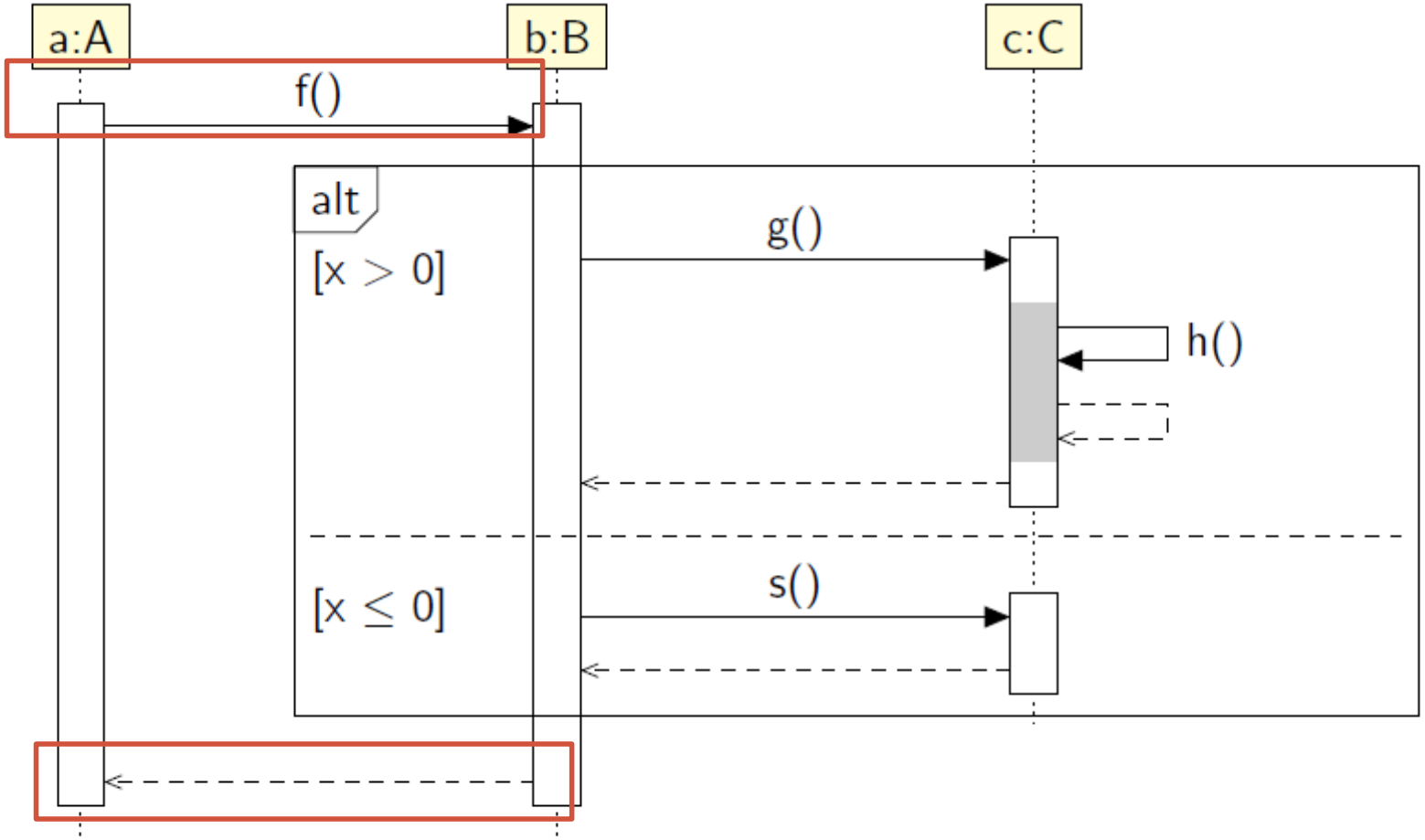
- Each participant is represented in a separate class.
- Any class that receives a call must have the corresponding methods.
- Any method from which a call is made must have the corresponding code to make that call.
- The indicated loop (loop) must be in the correct method of the correct class.
- The indicated condition (alt/opt) must be in the correct method of the correct class.

Write the minimal Java code for the classes `A`, `B` and `C` that captures all the behaviour in the following UML Sequence Diagram. The bodies of all methods other than `f()`, `g()`, `h()` and `s()` should be left empty. Constructors and properties (i.e. get and set methods) for the classes do not need to be shown.



```
public class A
{
    public void someMethod()
    {   ...
        result = b.f();
        ...
    }
}

public class B
{
    public SomeType1 f()
    {   ...
        if (x > 0)
        { ...    result1 = c.g(); ... }
        else
        { ...    result2 = c.s(); ... }
        ...
    }
}

public class C
{
    public SomeType2 g()
    {   ...
        result = this.h();
        ...
    }

    private SomeType3 h()
    {   ... }

    public SomeType4 s()
    {   ... }
}
```

Write the minimal Java code for the classes `A`, `B` and `C` that captures all the behaviour in the following UML Sequence Diagram. The bodies of all methods other than `f()`, `g()`, `h()` and `s()` should be left empty. Constructors and properties (i.e. get and set methods) for the classes do not need to be shown.

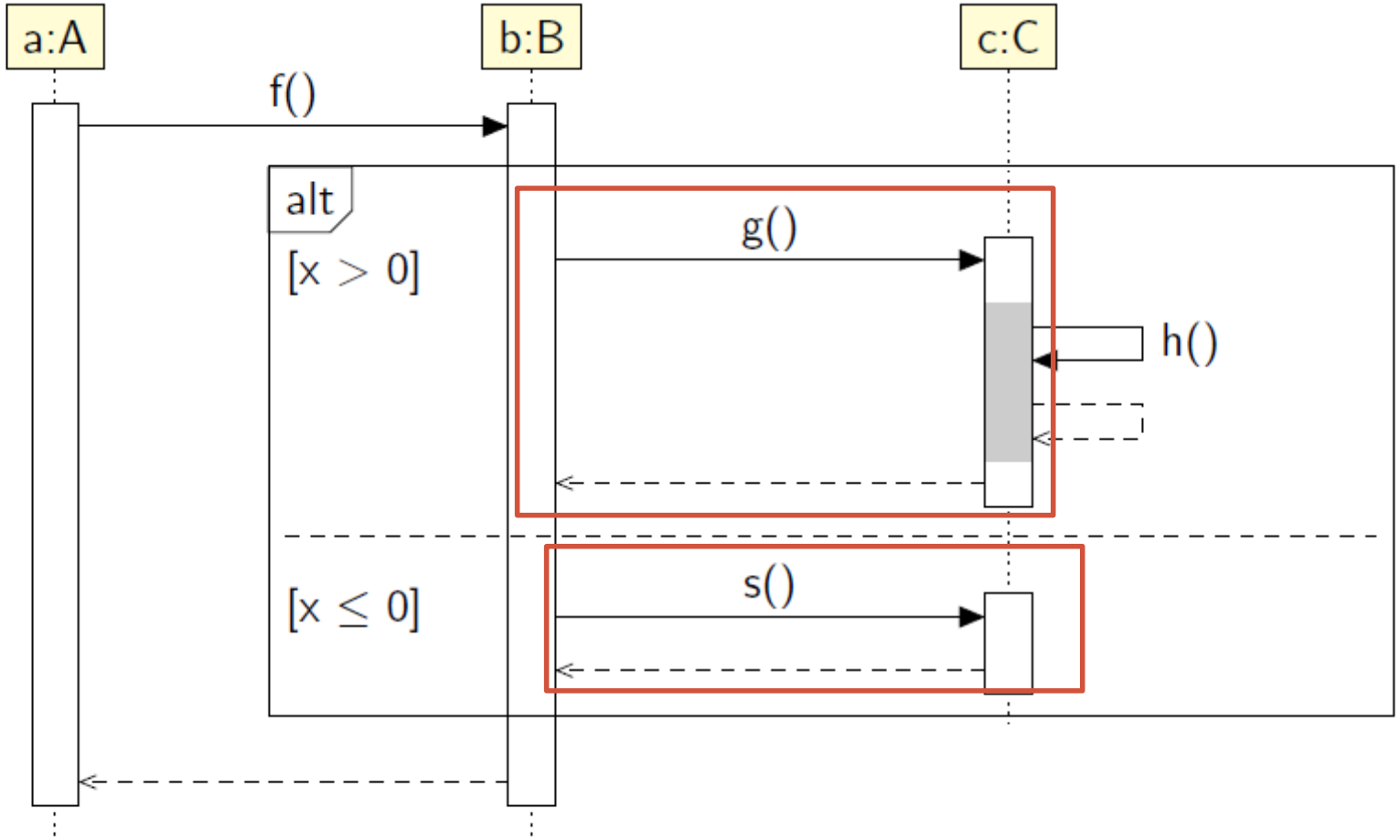


```
public class A
{
    public void someMethod()
    {   ...
        result = b.f();
        ...
    }
}
```

```
public class B
{
    public SomeType1 f()
    {   ...
        if (x > 0)
        { ...    result1 = c.g(); ... }
        else
        { ...    result2 = c.s(); ... }
        ...
    }
}
```

```
public class C
{
    public SomeType2 g()
    {   ...
        result = this.h();
        ...
    }

    private SomeType3 h()
    {   ... }

    public SomeType4 s()
    {   ... }
}
```

Write the minimal Java code for the classes `A`, `B` and `C` that captures all the behaviour in the following UML Sequence Diagram. The bodies of all methods other than `f()`, `g()`, `h()` and `s()` should be left empty. Constructors and properties (i.e. get and set methods) for the classes do not need to be shown.

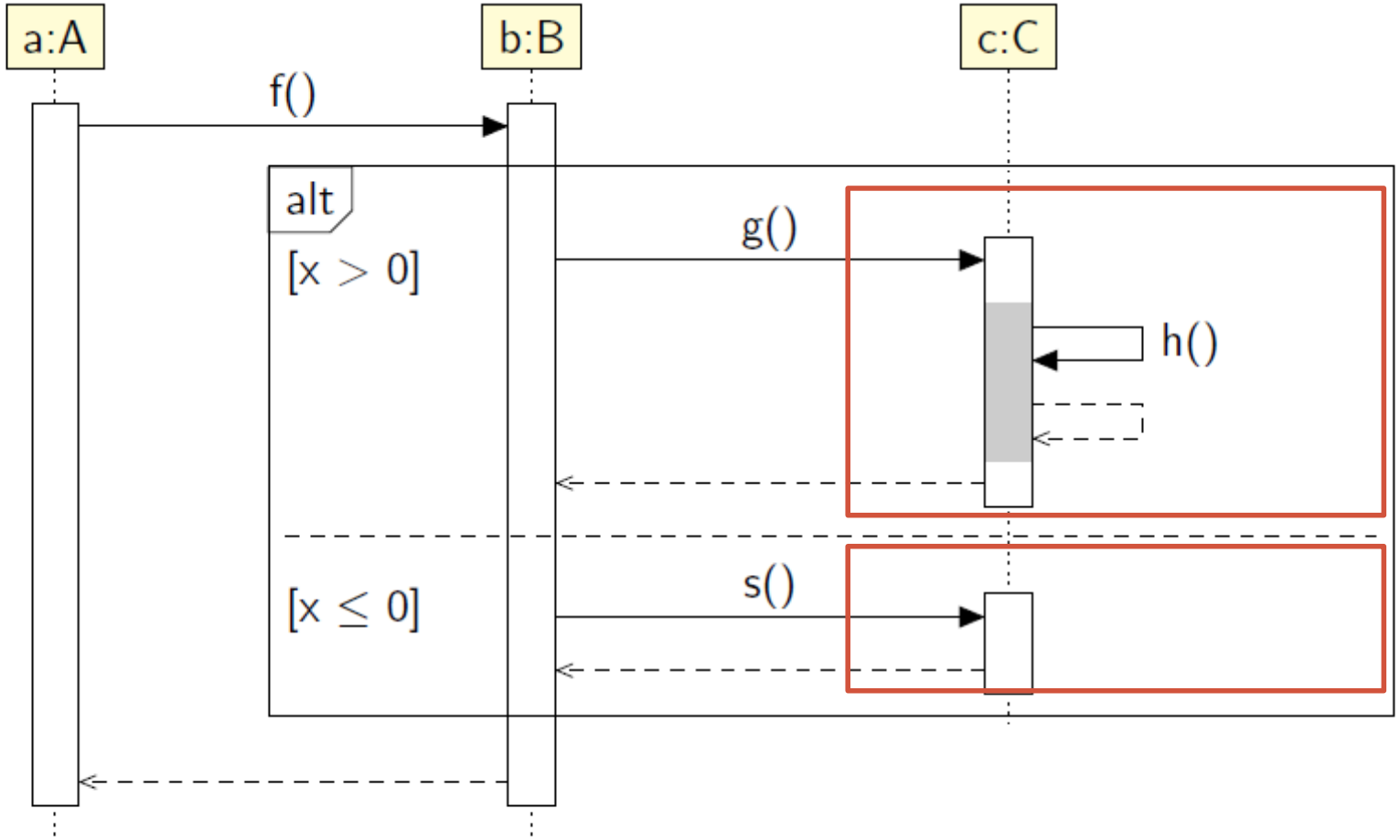


```
public class A
{
    public void someMethod()
    {   ...
        result = b.f();
        ...
    }
}
```

```
public class B
{
    public SomeType1 f()
    {   ...
        if (x > 0)
        { ...    result1 = c.g(); ... }
        else
        { ...    result2 = c.s(); ... }
        ...
    }
}
```

```
public class C
{
    public SomeType2 g()
    {   ...
        result = this.h();
        ...
    }

    private SomeType3 h()
    {   ... }

    public SomeType4 s()
    {   ... }
}
```

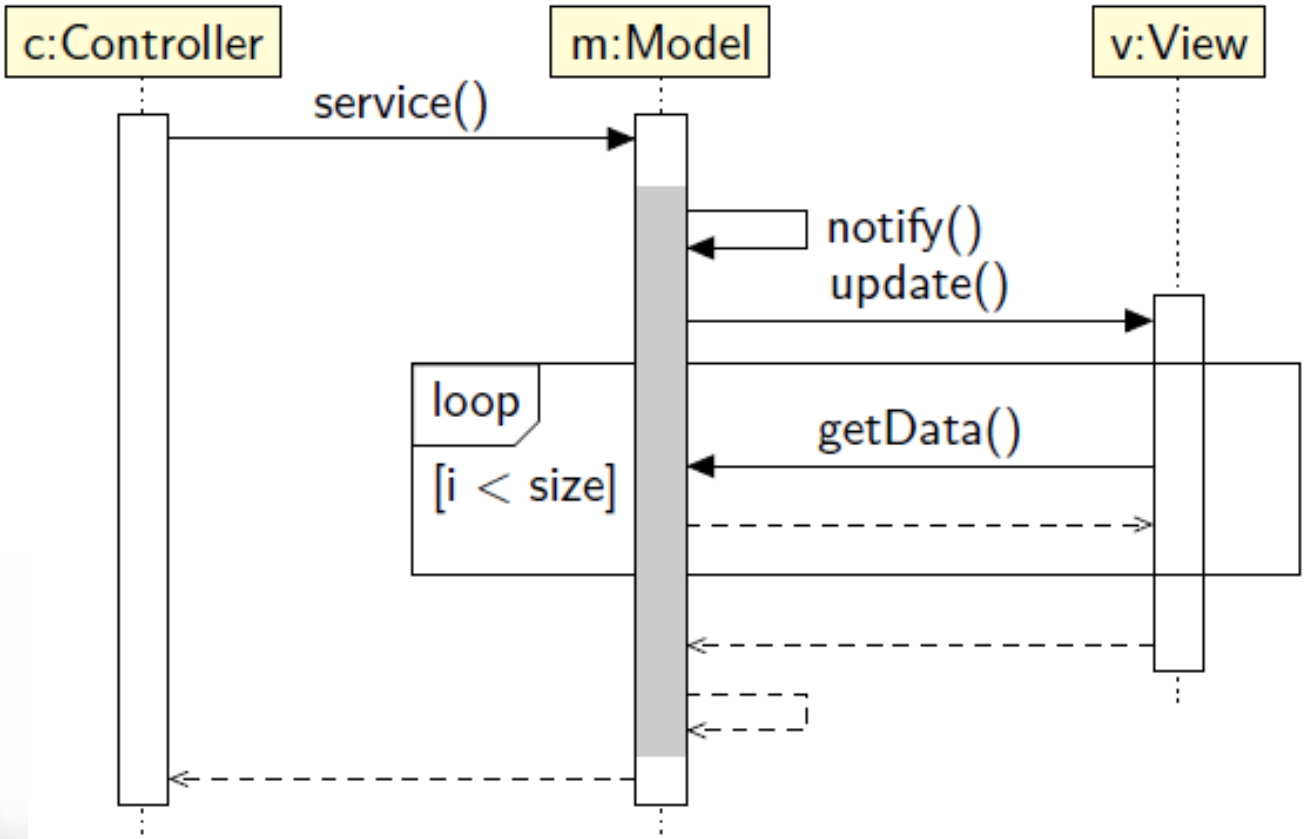
Write the minimal Java code for the classes `A`, `B` and `C` that captures all the behaviour in the following UML Sequence Diagram. The bodies of all methods other than `f()`, `g()`, `h()` and `s()` should be left empty. Constructors and properties (i.e. get and set methods) for the classes do not need to be shown.



```
public class A
{
    public void someMethod()
    {   ...
        result = b.f();
        ...
    }
}

public class B
{
    public SomeType1 f()
    {   ...
        if (x > 0)
        { ...    result1 = c.g(); ... }
        else
        { ...    result2 = c.s(); ... }
        ...
    }
}
```

```
public class C
{
    public SomeType2 g()
    {   ...
        result = this.h();
        ...
    }

    private SomeType3 h()
    {   ... }

    public SomeType4 s()
    {   ... }
}
```

Write the minimal Java code for the classes `Controller`, `Model` and `View` that captures all the behaviour in the following UML Sequence Diagram. The bodies of all methods should be left empty other than for the calls indicated in the diagram. Constructors and properties (i.e. get and set methods) for the classes do not need to be shown.

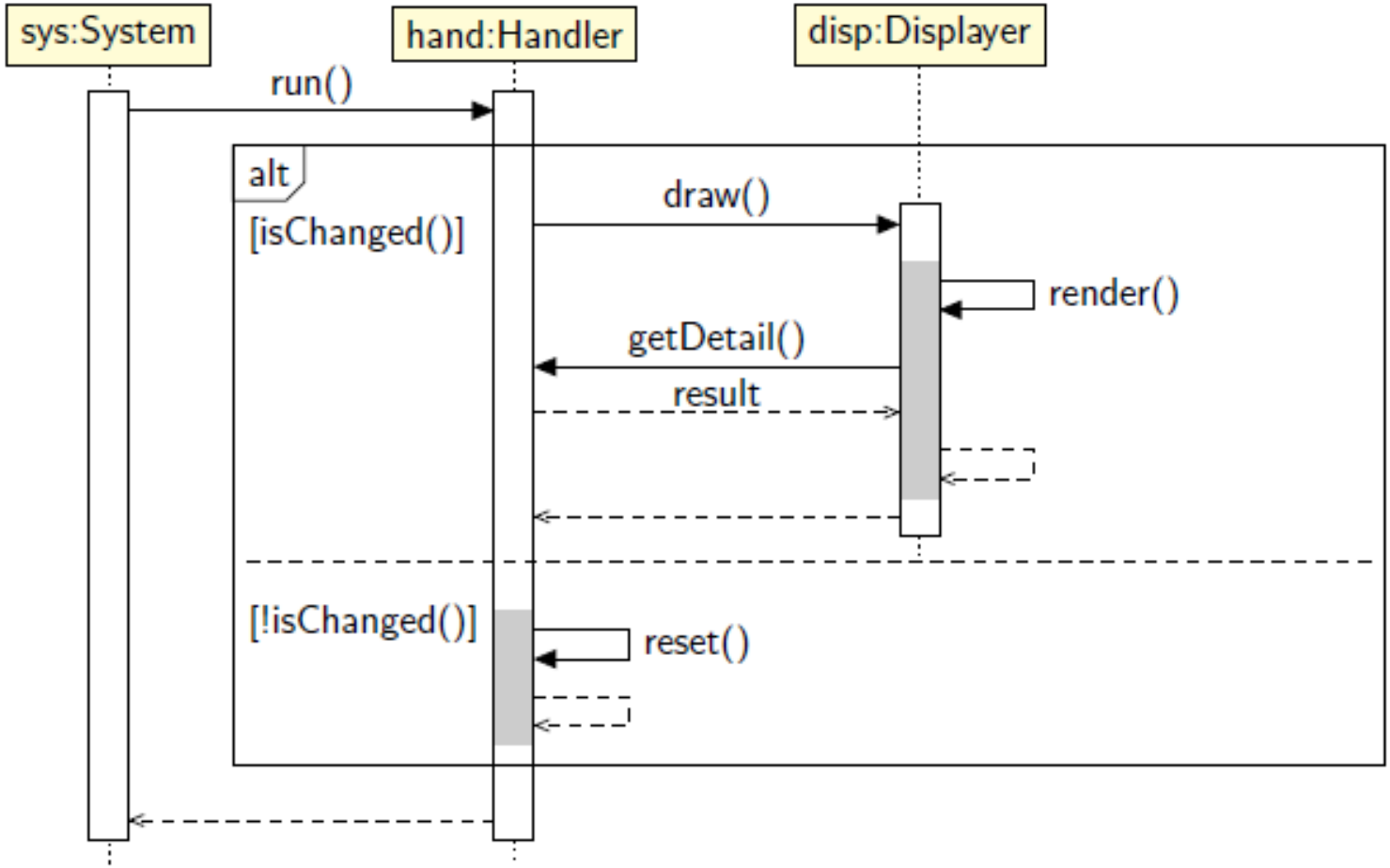


```
public class Controller
{
    public void someMethod()
    {   ...
        result = m.service();
        ...
    }
}

public class Model
{
    public SomeType1 service()
    {   ...
        result = this.notify();
        ...
    }

    public SomeType2 notify()
    {   ...
        result = v.update();
        ...
    }

    public SomeType3 getData();
    { ... }
}
```

```
public class View
{
    public SomeType4 update()
    {   ...
        while (i < size)
        {   ...
            result = m.getData();
            ...
        }
        ...
    }
}
```

Exercise

Write the minimal Java code for the classes `System`, `Handler` and `Displayer` that captures all the behaviour in the following UML Sequence Diagram. The bodies of all methods should be left empty other than for the calls indicated in the diagram. Constructors and properties (i.e. get and set methods) for the classes do not need to be shown.



References

- A number of slides in this talk is based on:
  - Alan P. Sexton hand-outs (Introduction to Software Engineering. The University of Birmingham. Spring Semester 2014)

Thank YOU ☺