EXPERIENCE-BASED MODEL REFINEMENT

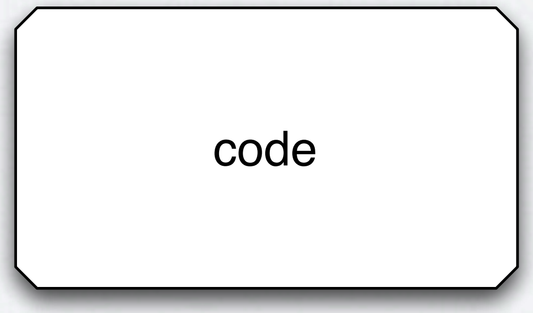
Alexis Marechal

SVARM workshop

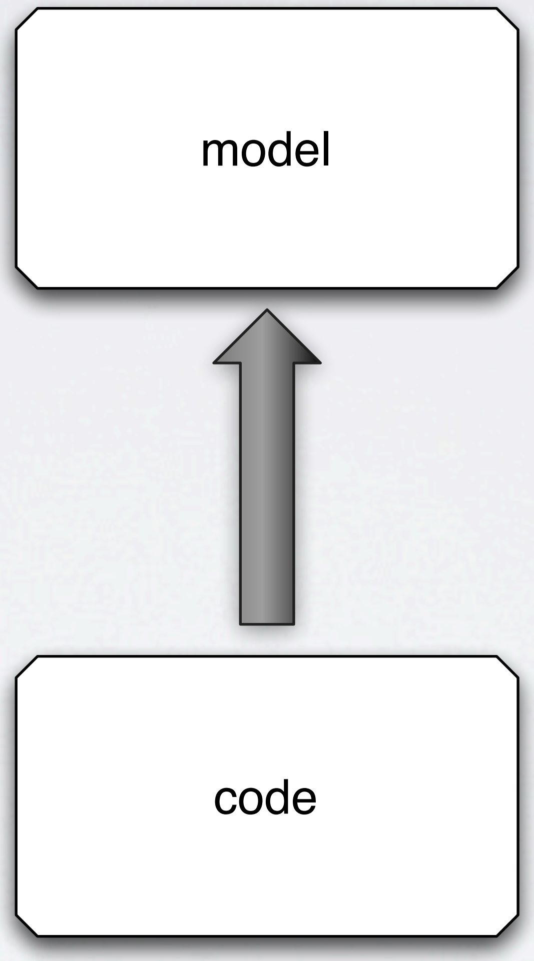
3/04/2011

G. Holzmann

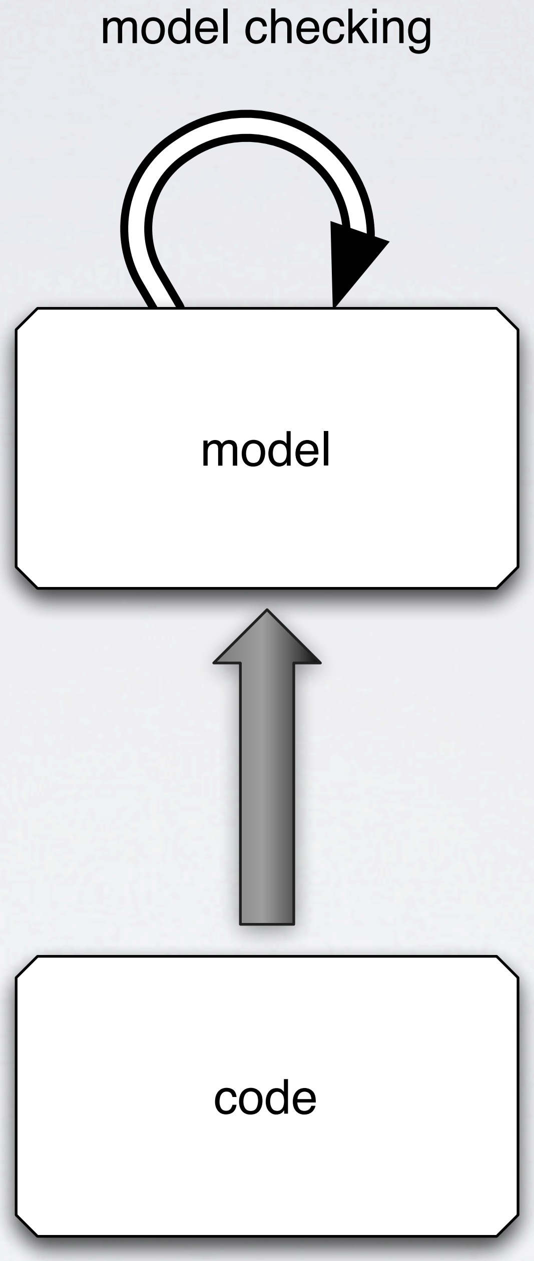
G. Holzmann



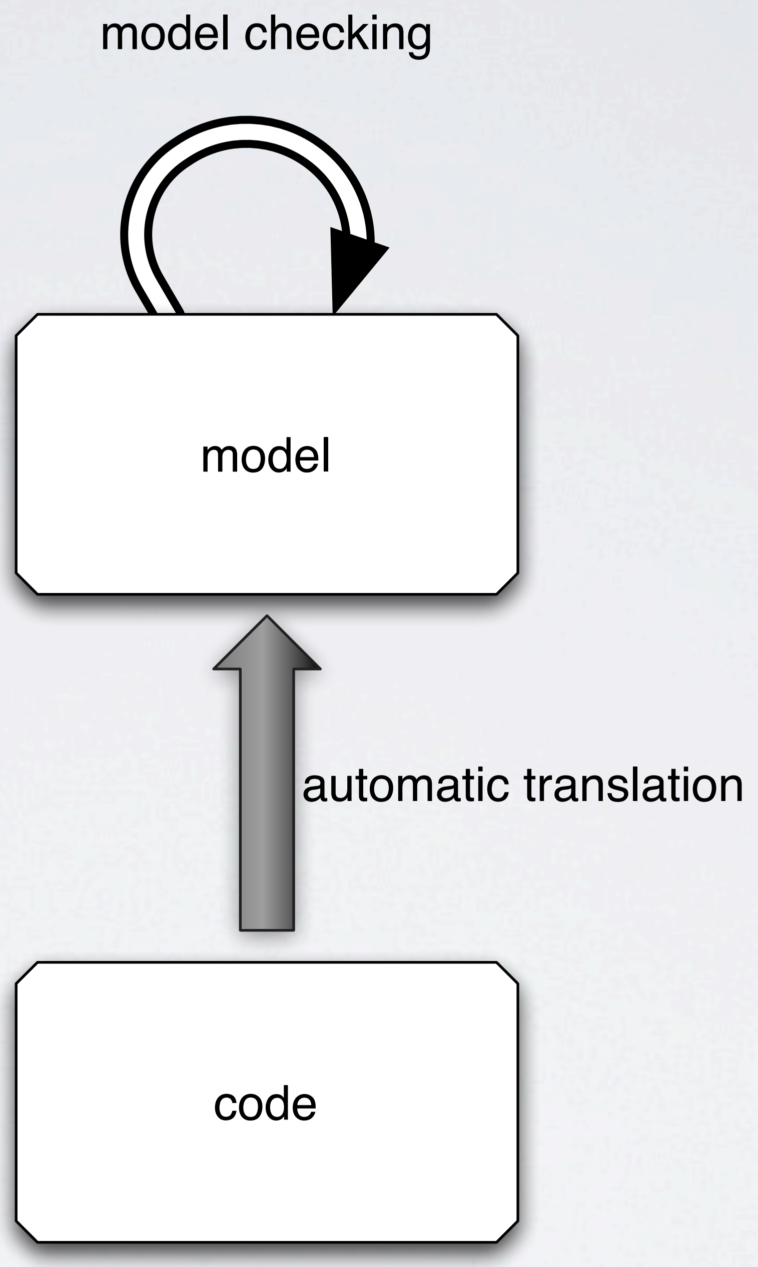
G. Holzmann



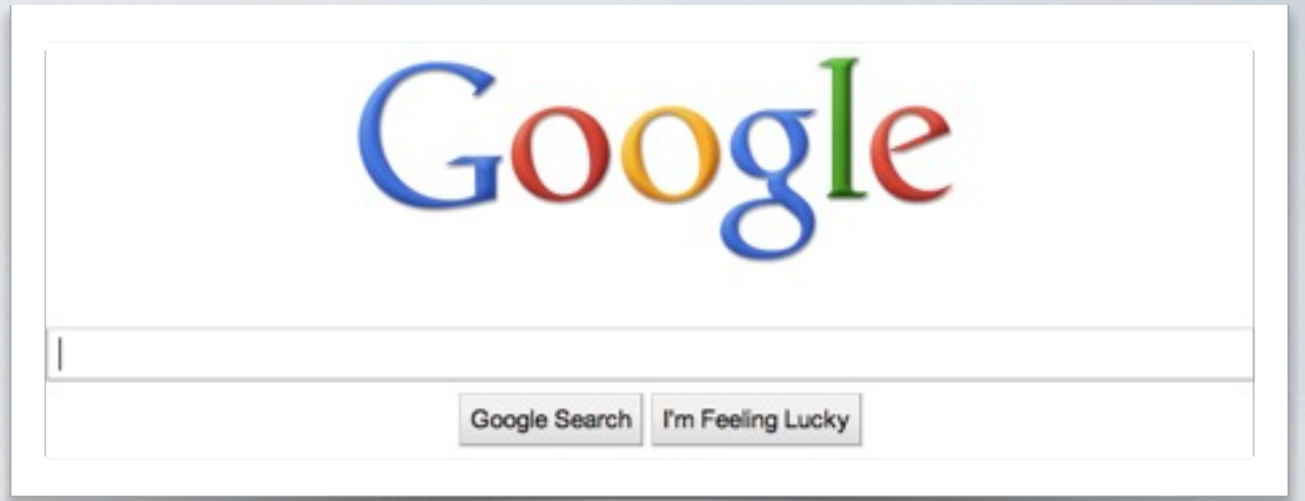
G. Holzmann

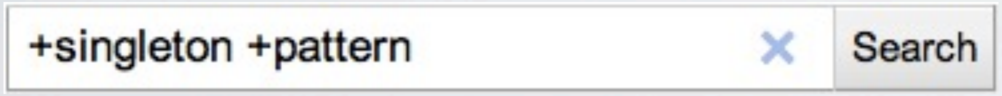


G. Holzmann

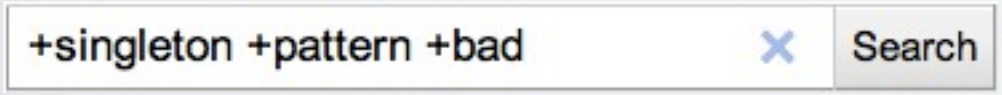


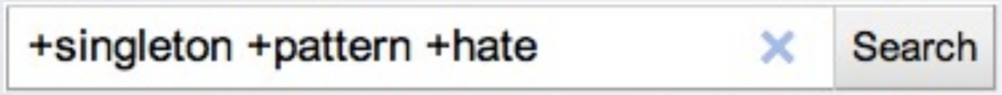
Singleton design pattern

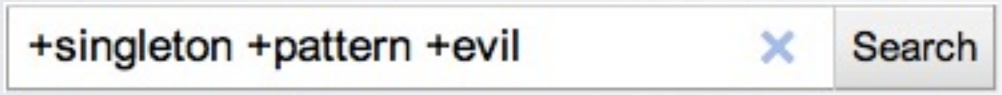


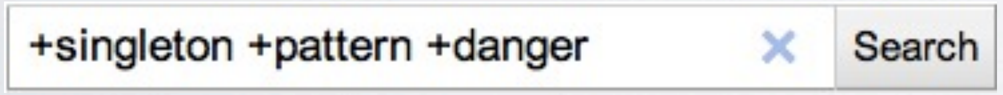

+singleton +pattern

760 000


+singleton +pattern +bad


+singleton +pattern +hate


+singleton +pattern +evil


+singleton +pattern +danger

230 600

Difficult tests

Code maintenance is harder

Creating sub-classes is difficult

Hidden dependencies

Difficult tests

Code maintenance is harder

EXTREMELY error prone

Creating sub-classes is difficult

Hidden dependencies

Difficult tests

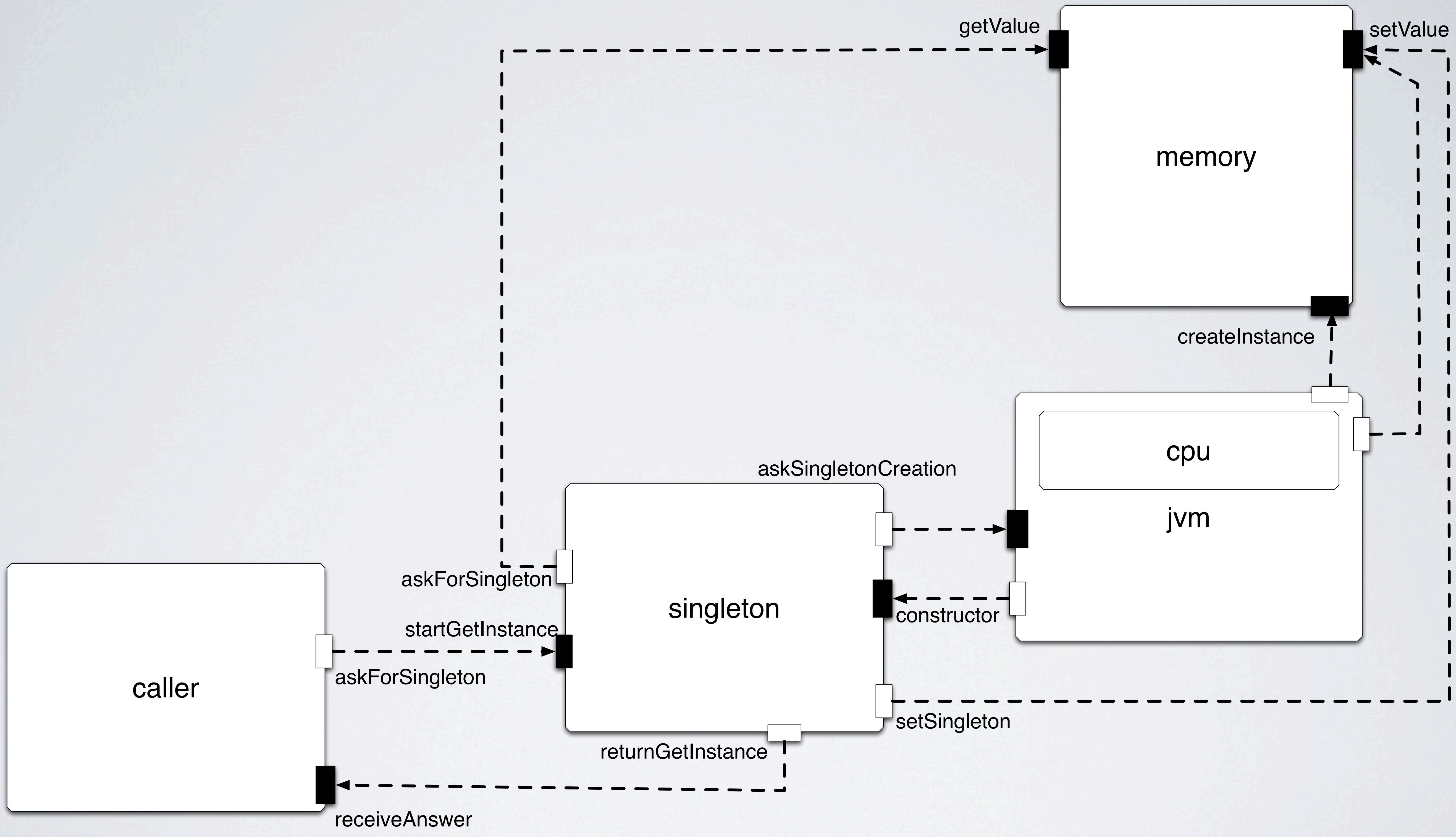
Code maintenance is harder

EXTREMELY error prone

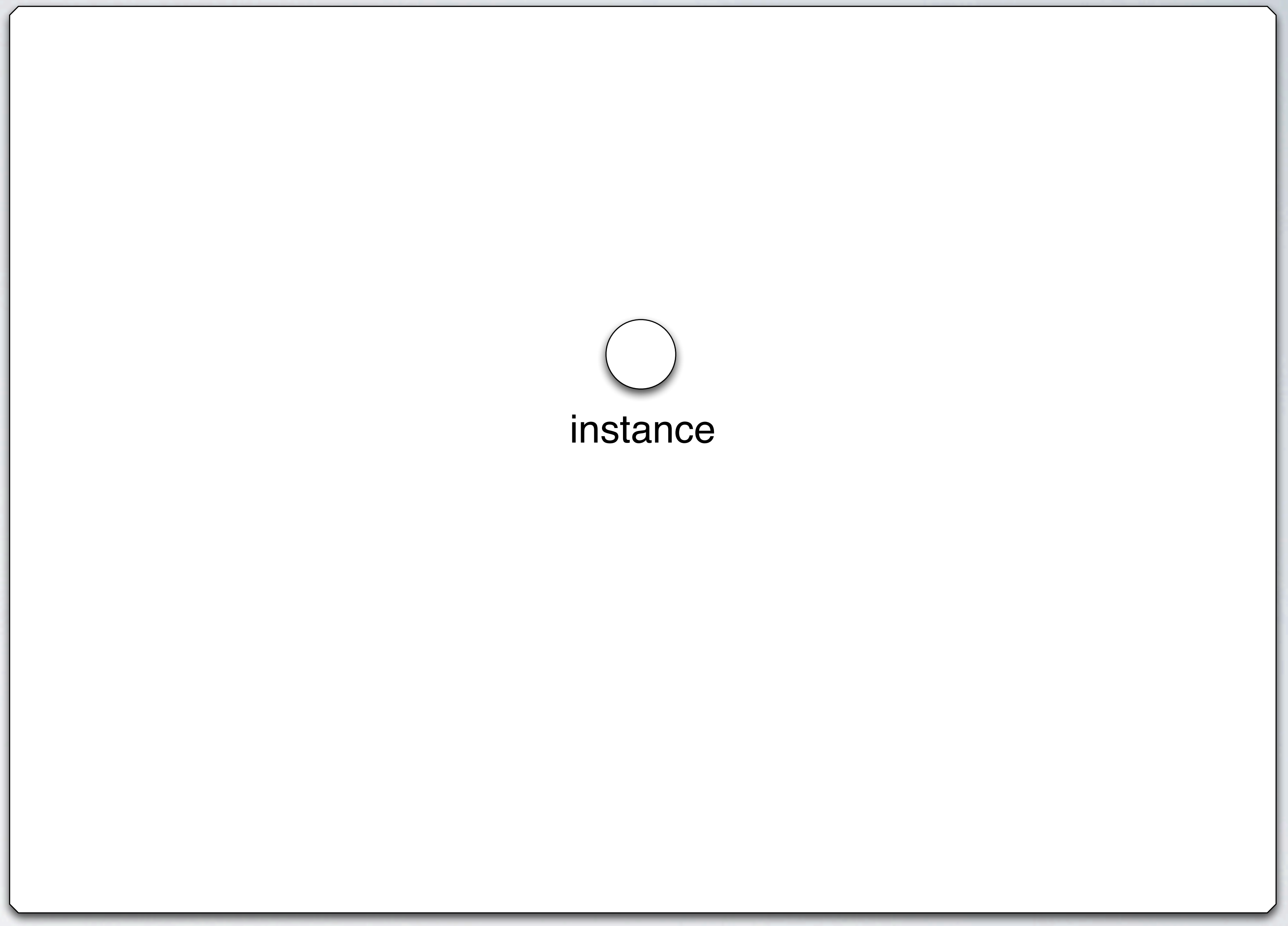
Creating sub-classes is difficult

Hidden dependencies

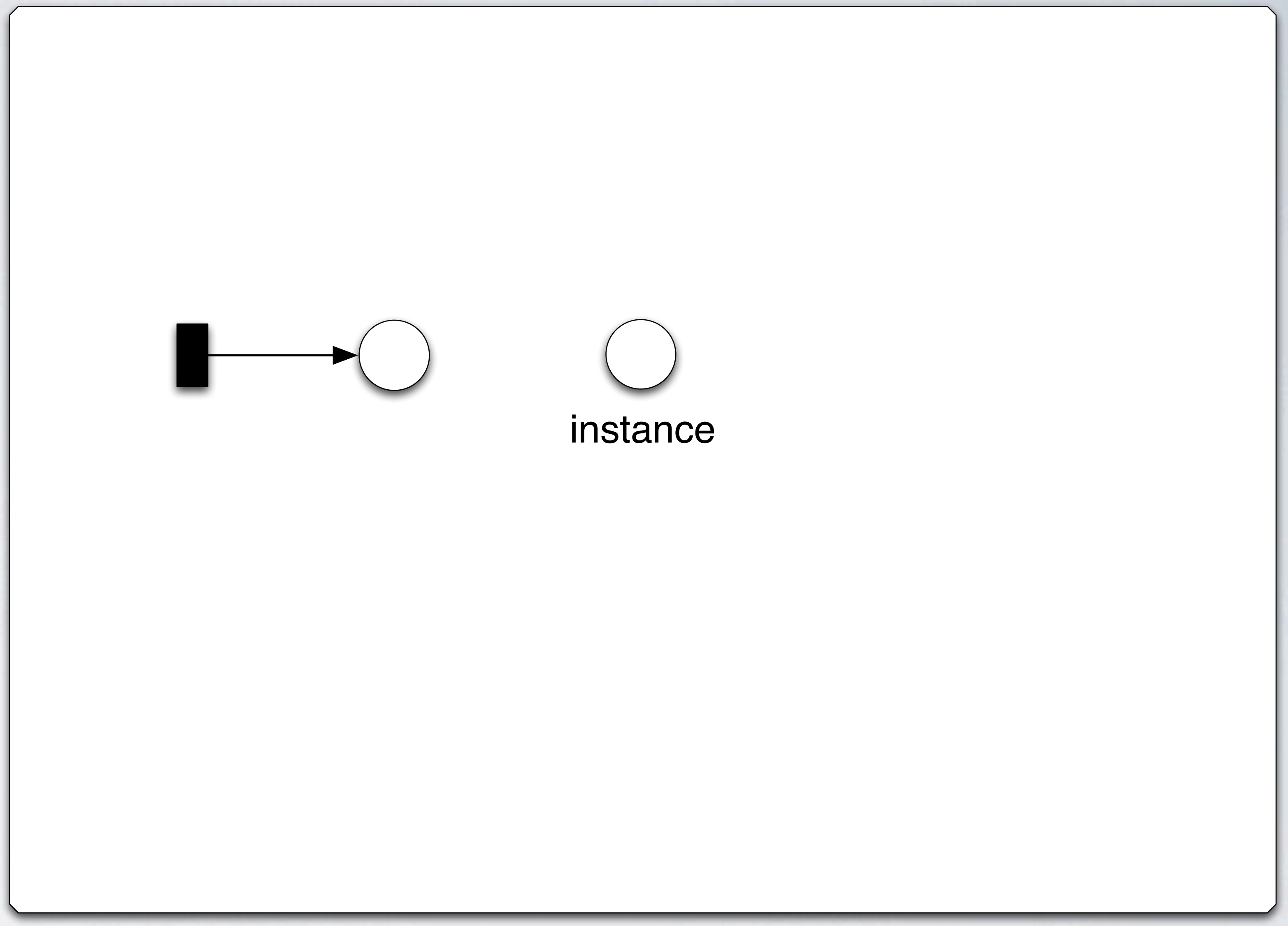
```java
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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


getValue
setValue
memory
createInstance
cpu
jvm
askSingletonCreation
askForSingleton
singleton
constructor
startGetInstance
caller
askForSingleton
setSingleton
returnGetInstance
receiveAnswer

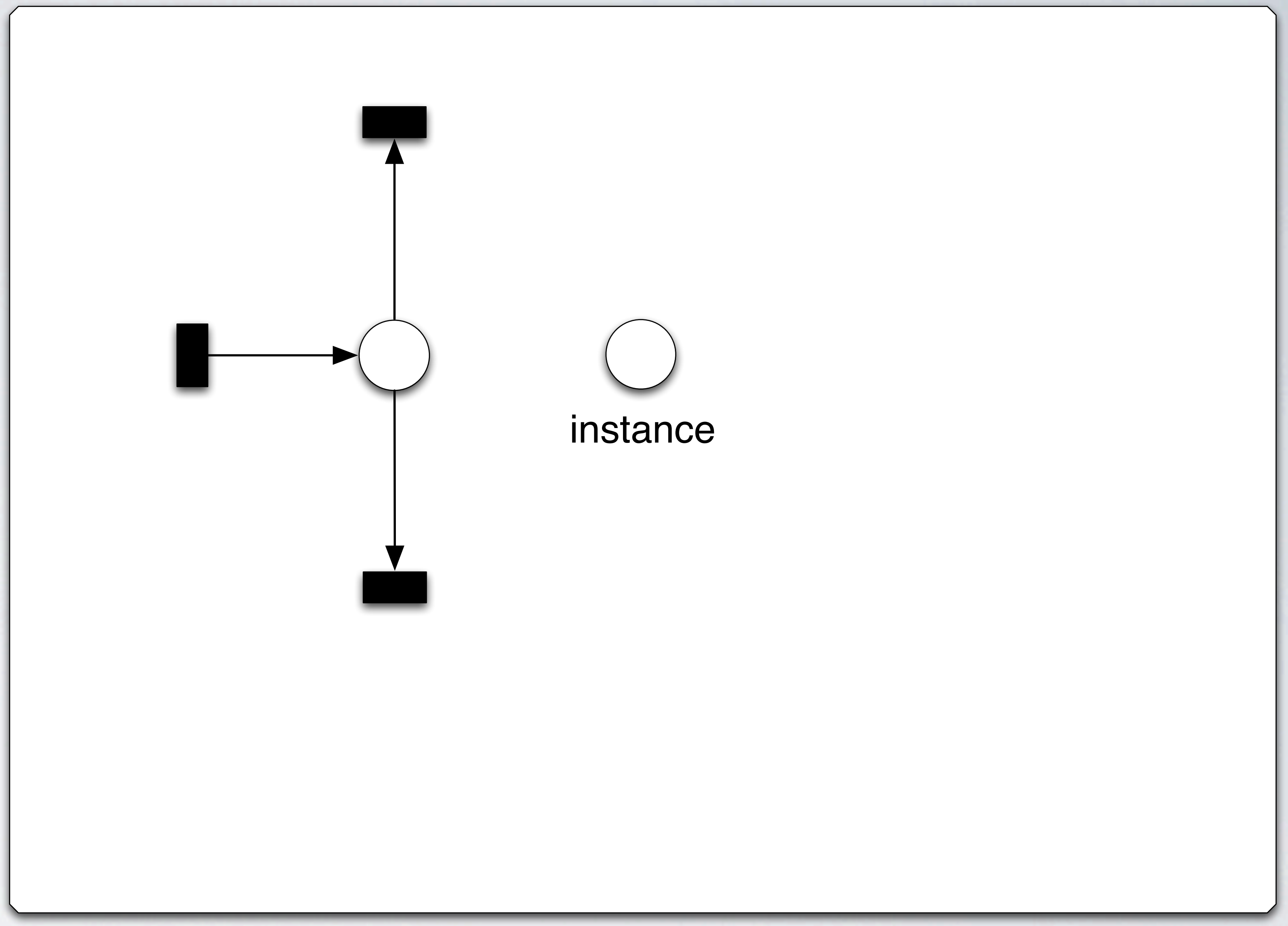
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



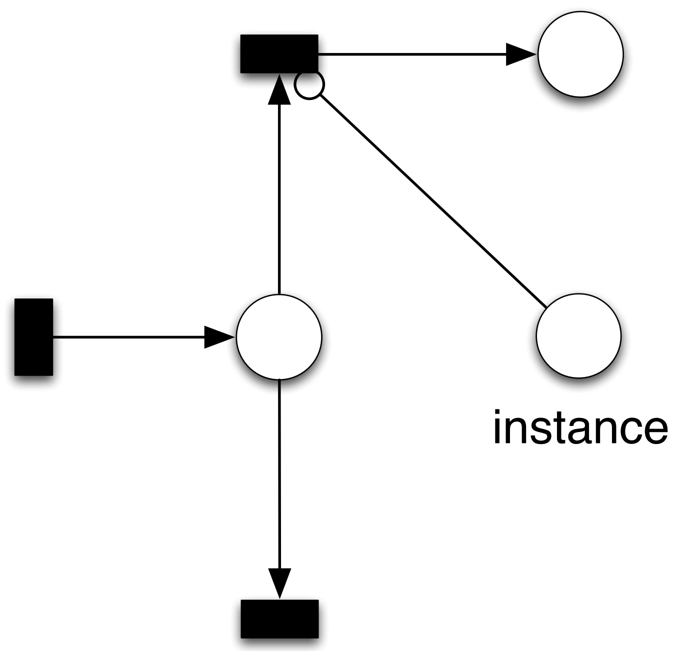
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



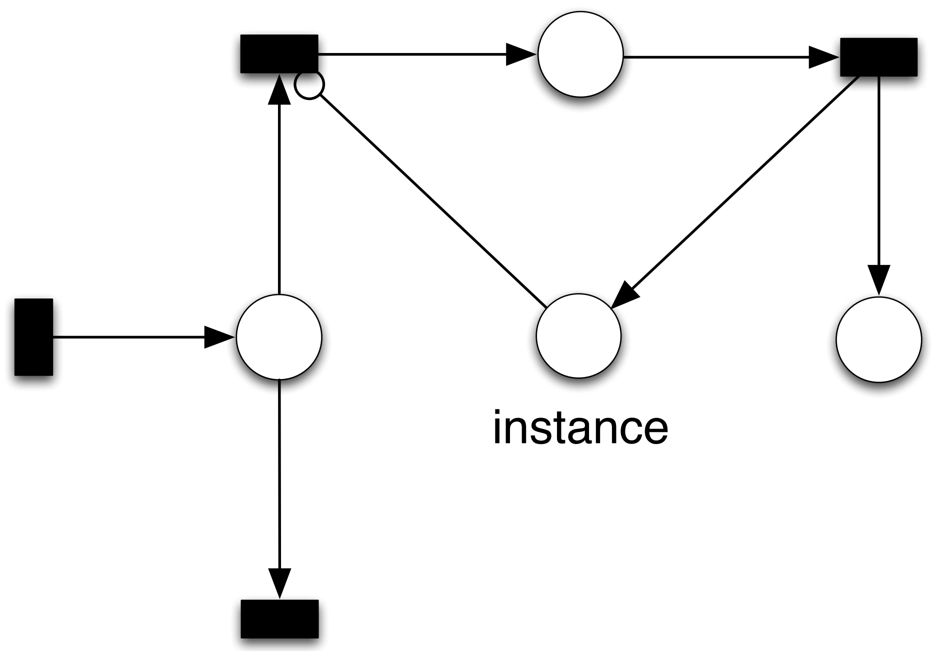
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



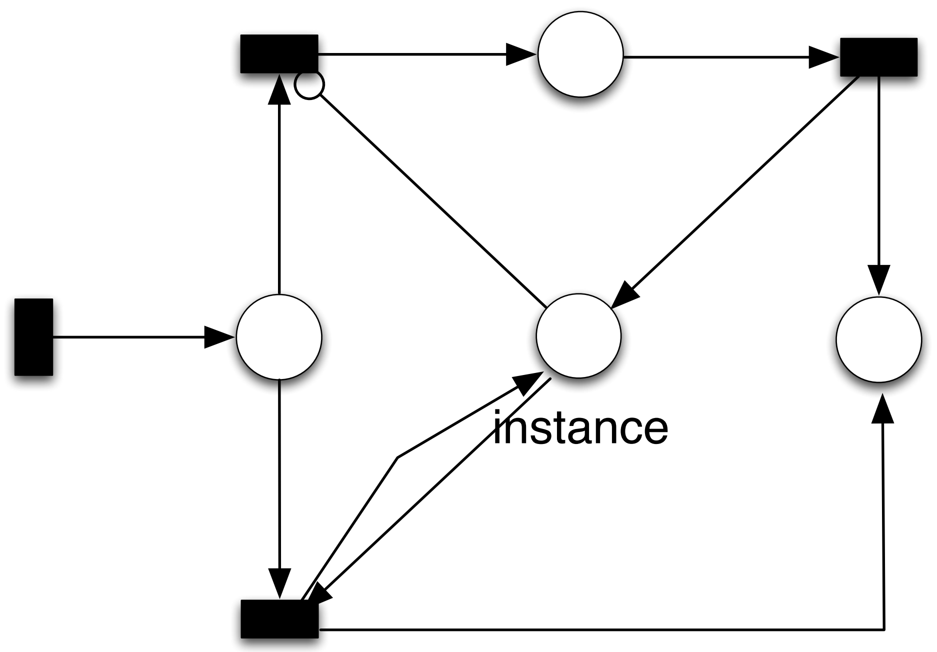
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



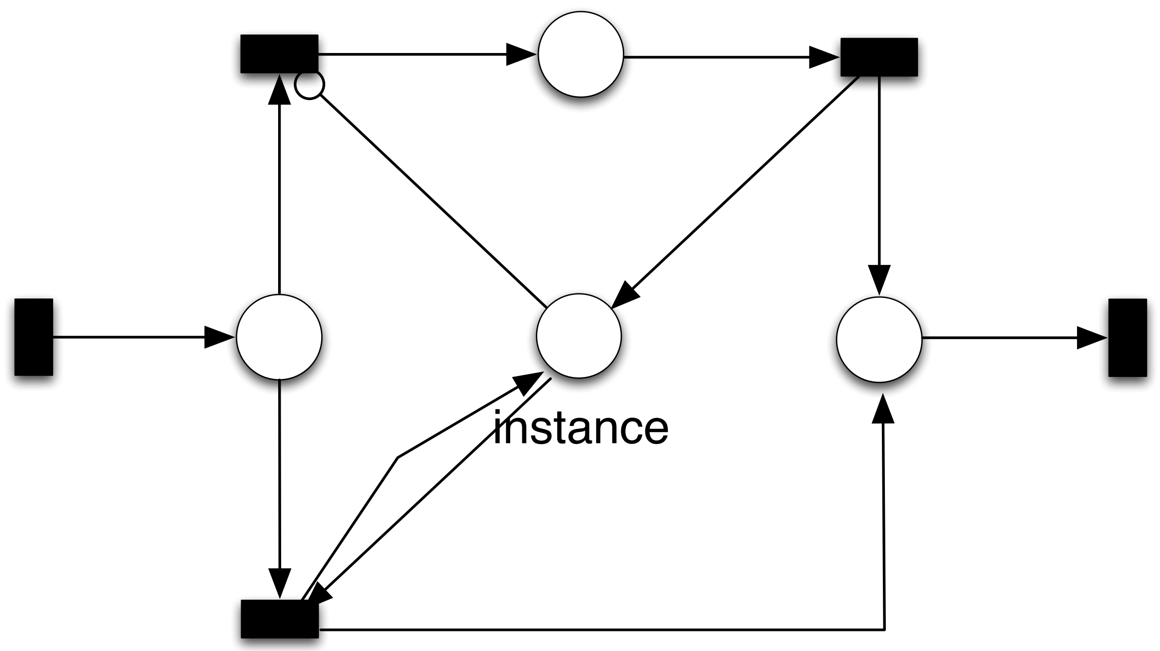
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



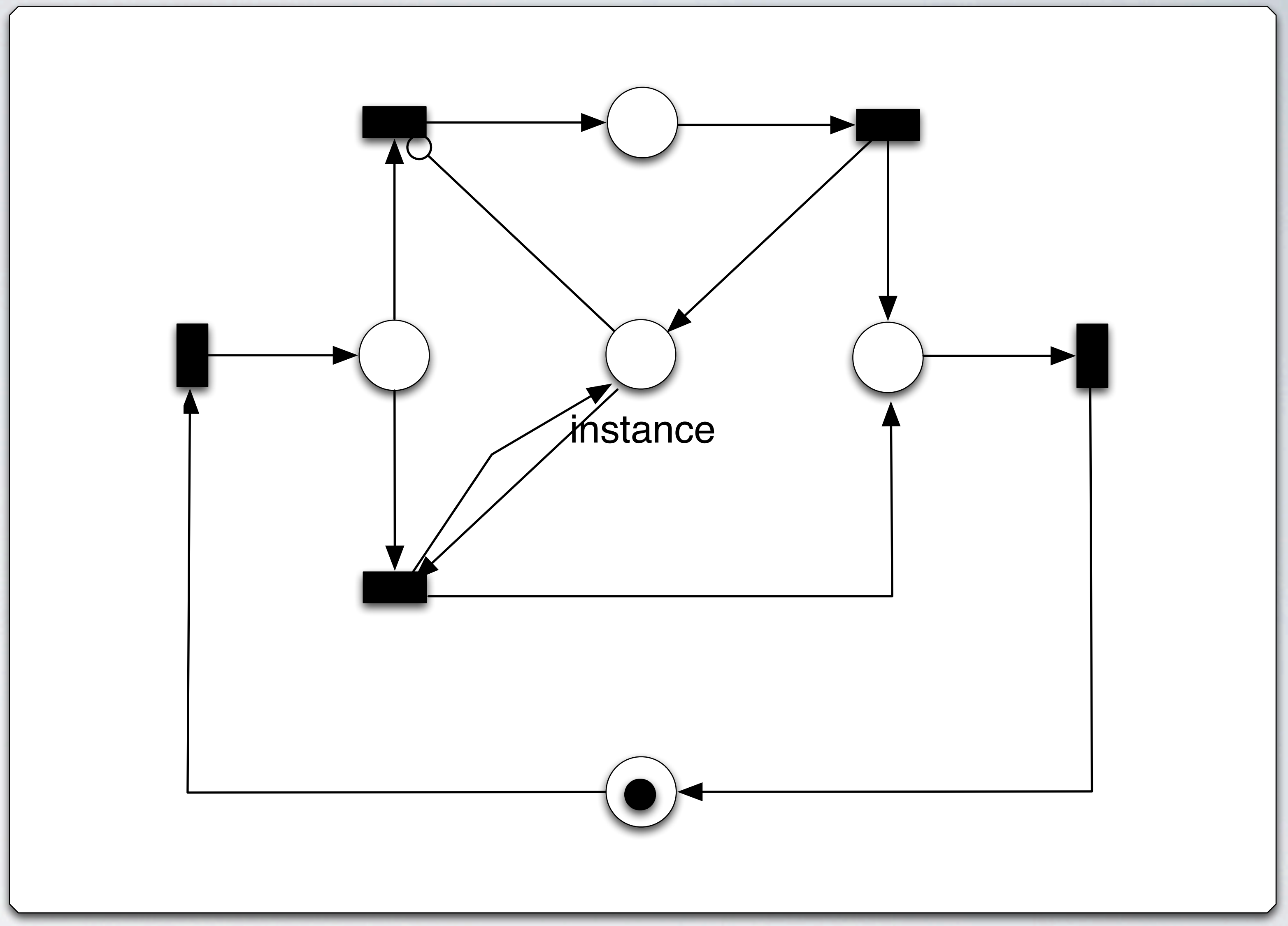
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



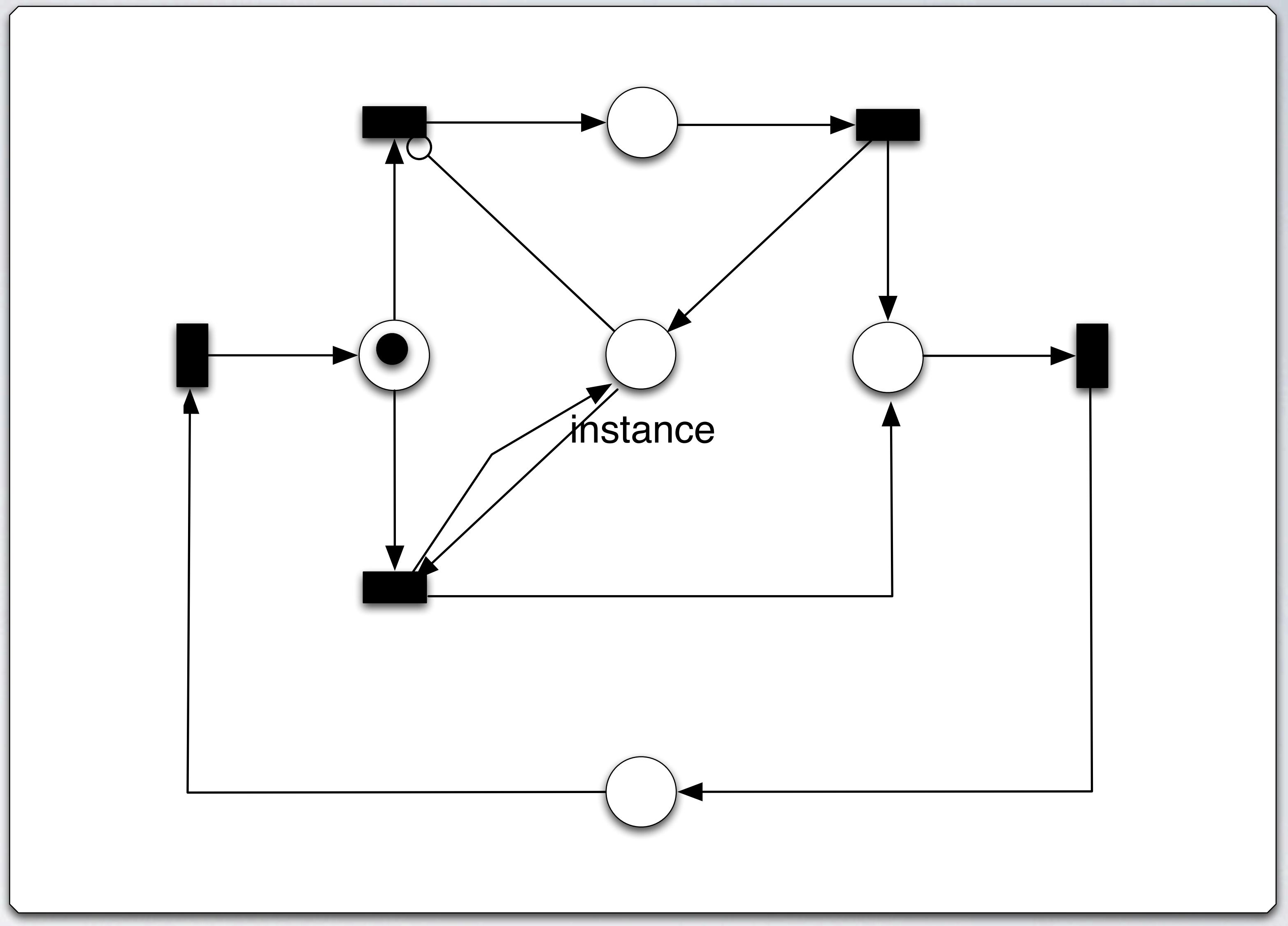
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



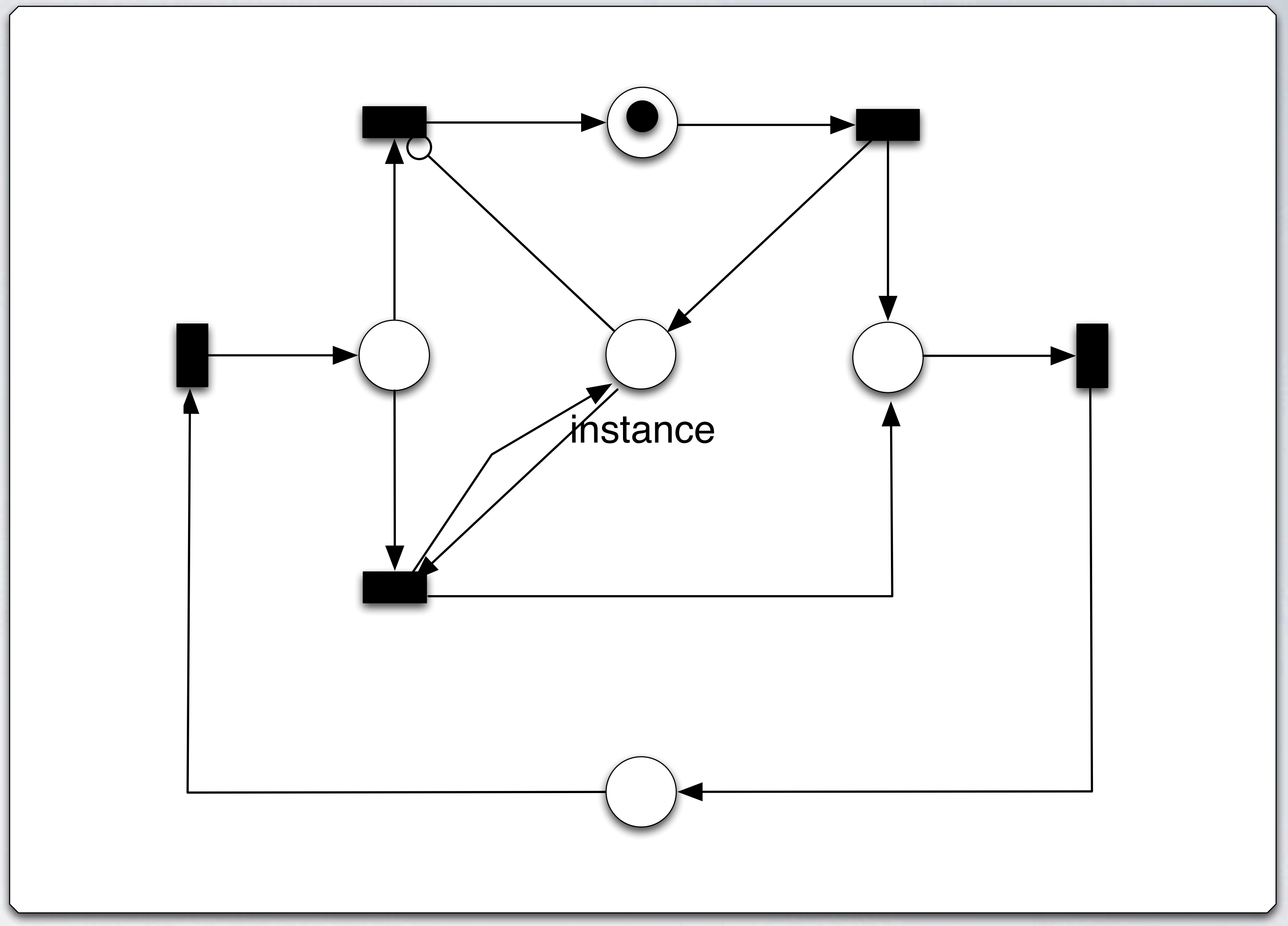
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



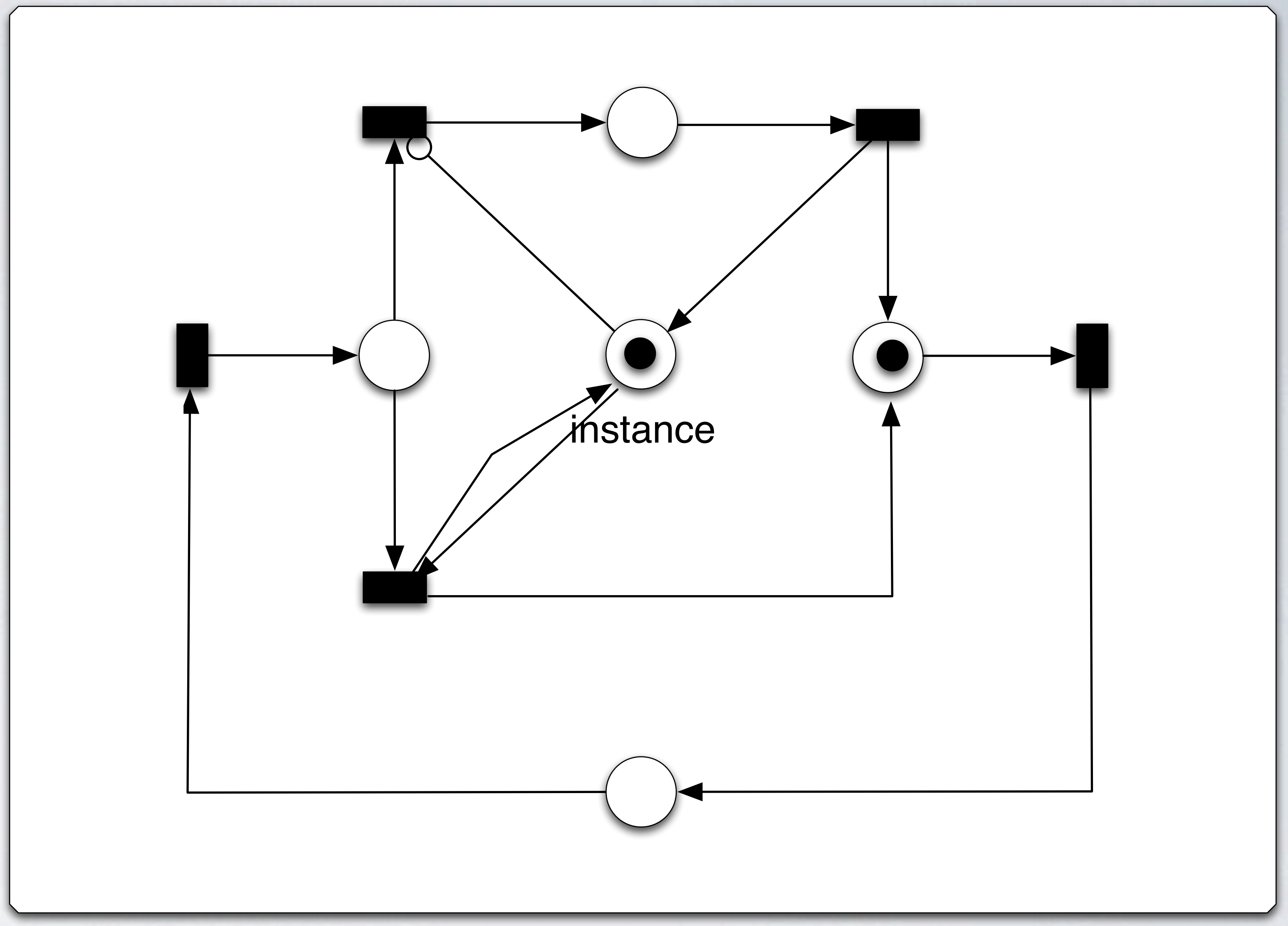
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



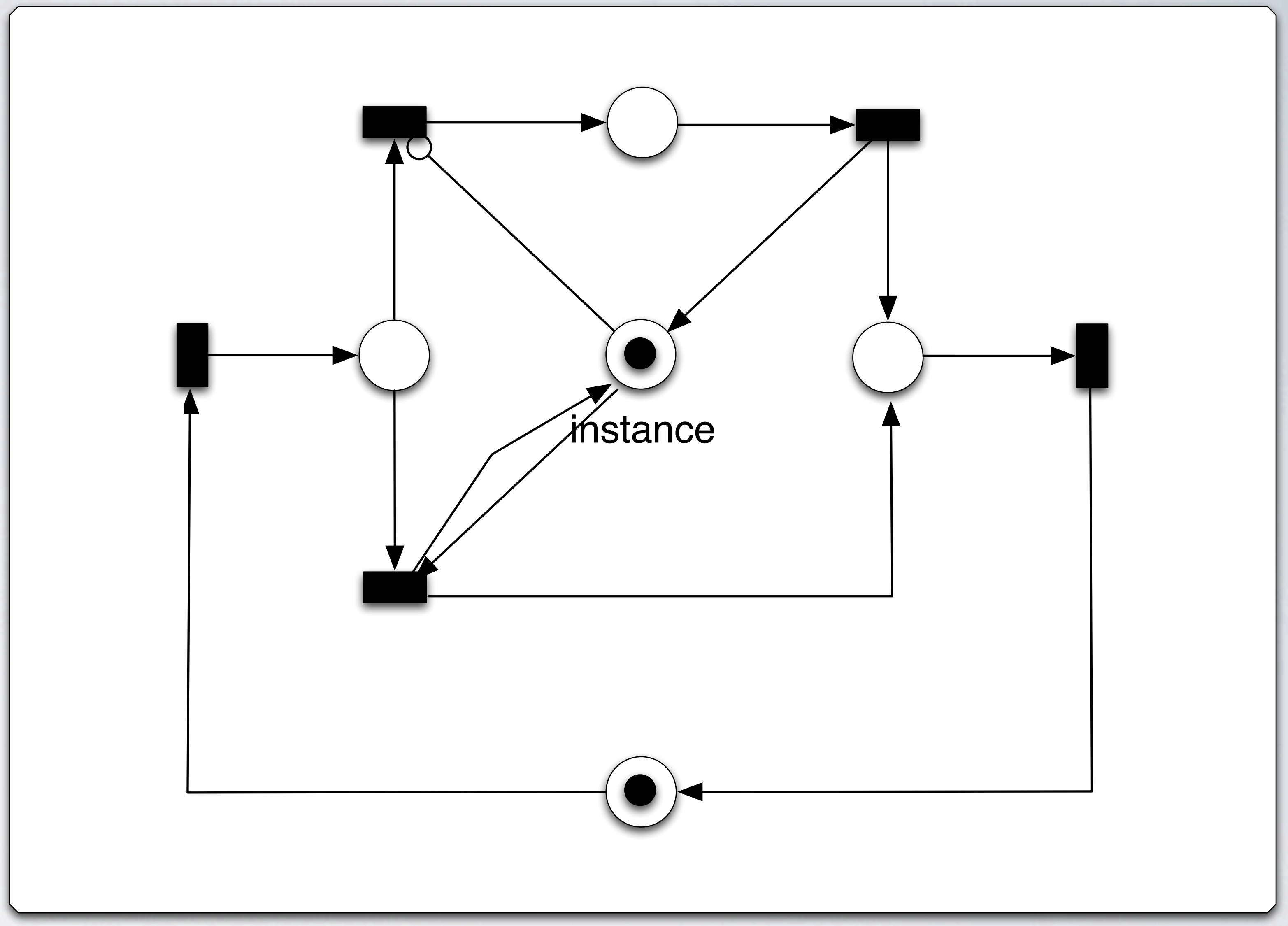
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



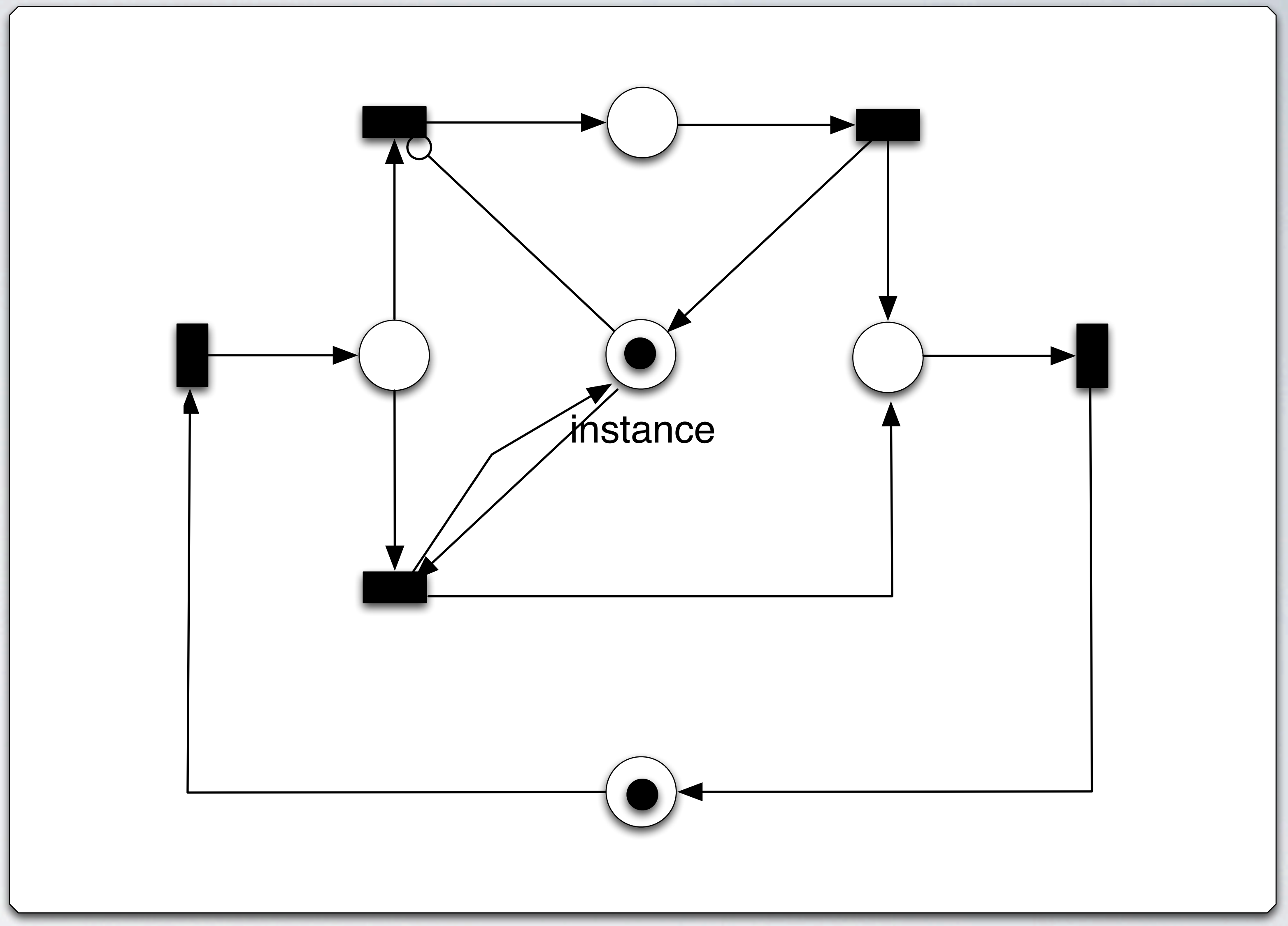
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



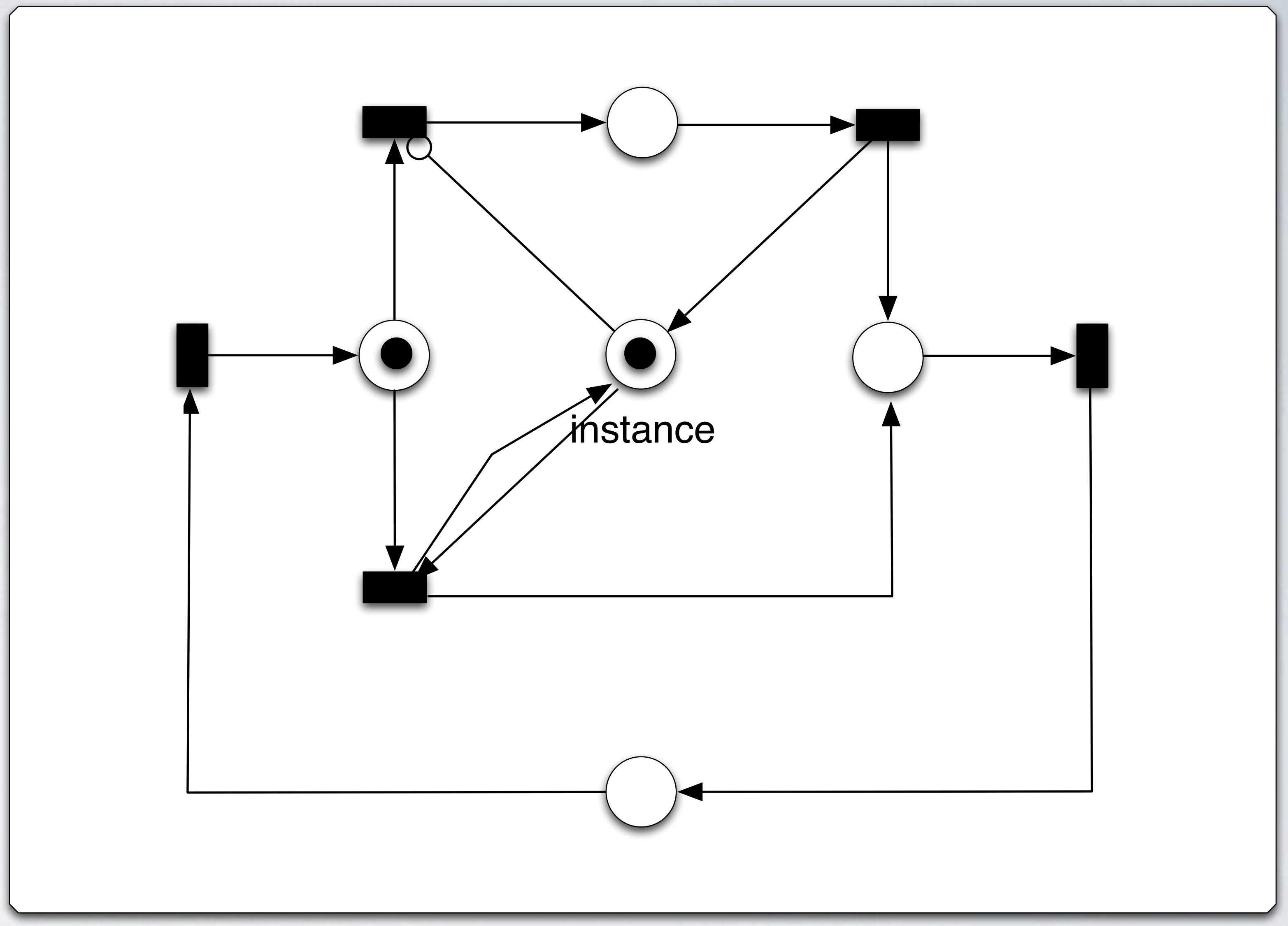
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



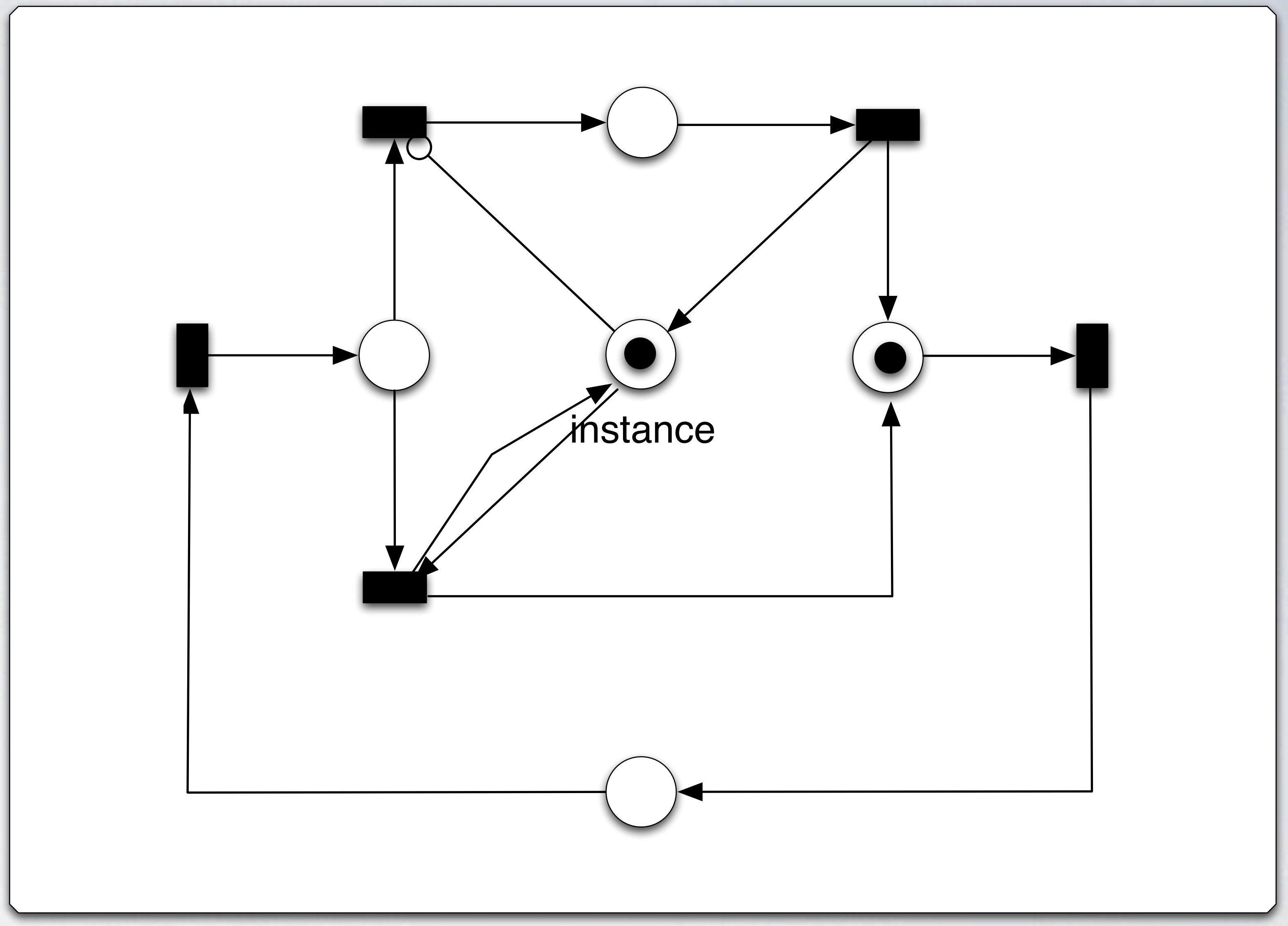
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



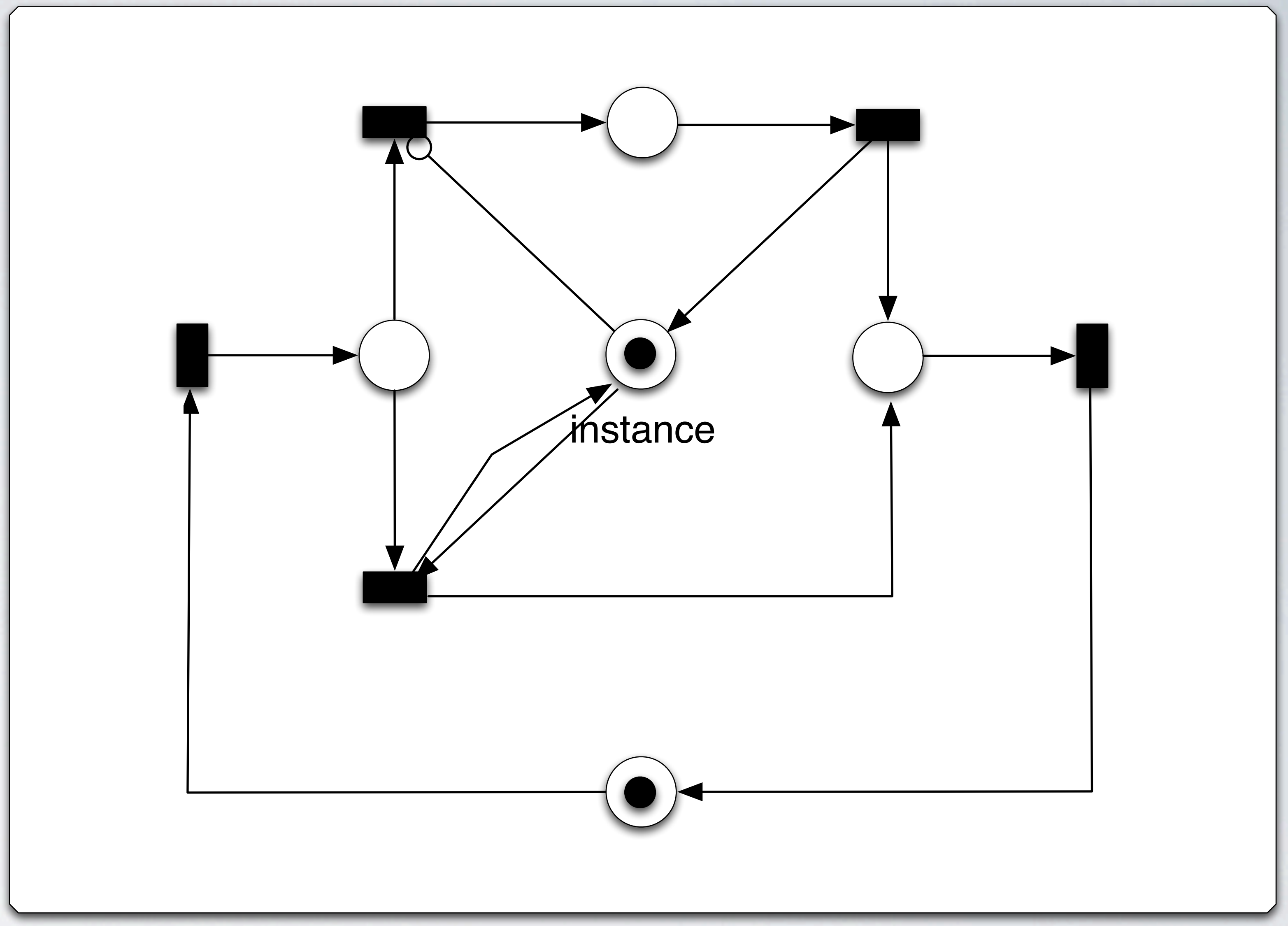
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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



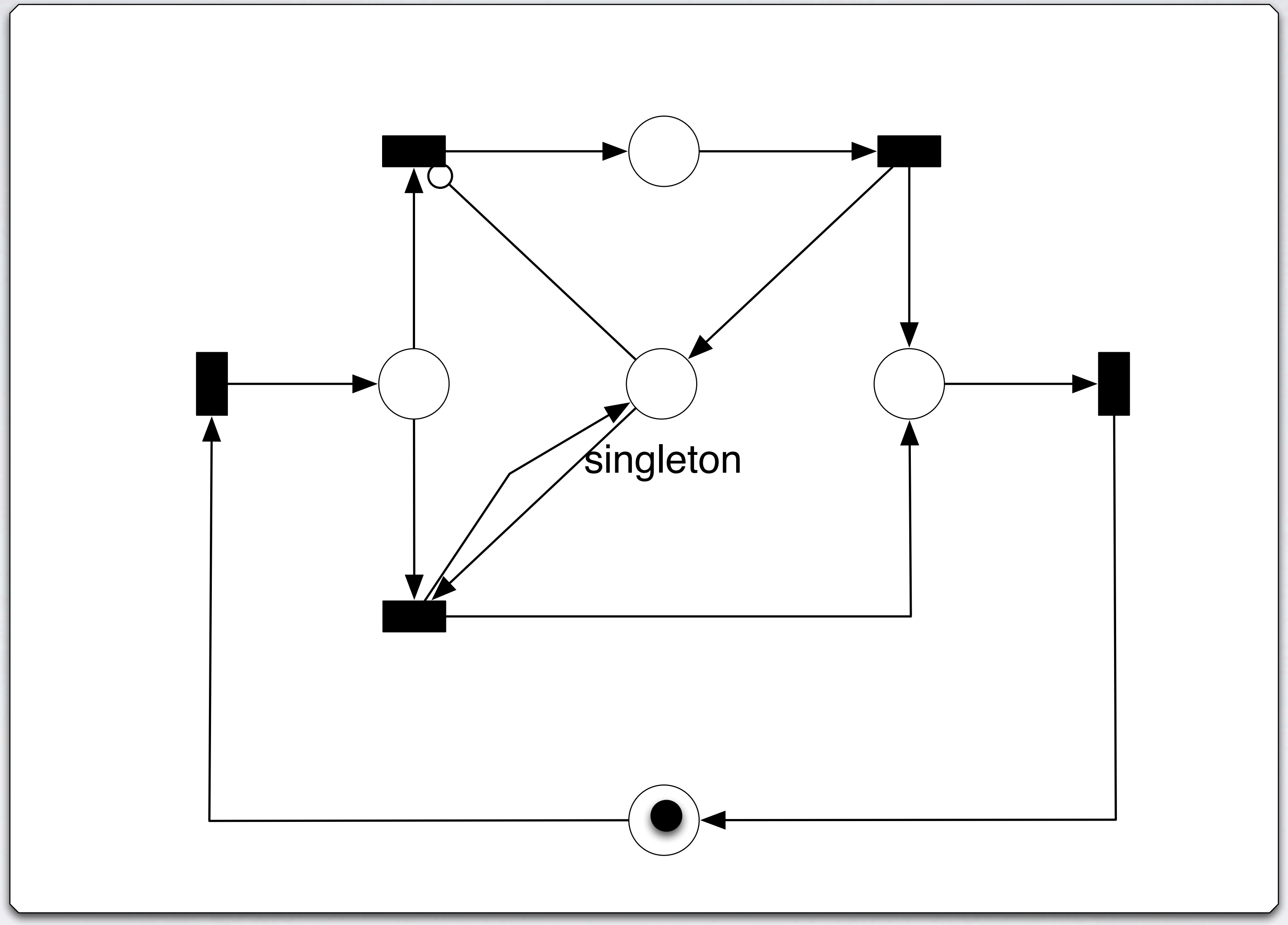
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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

It just works!

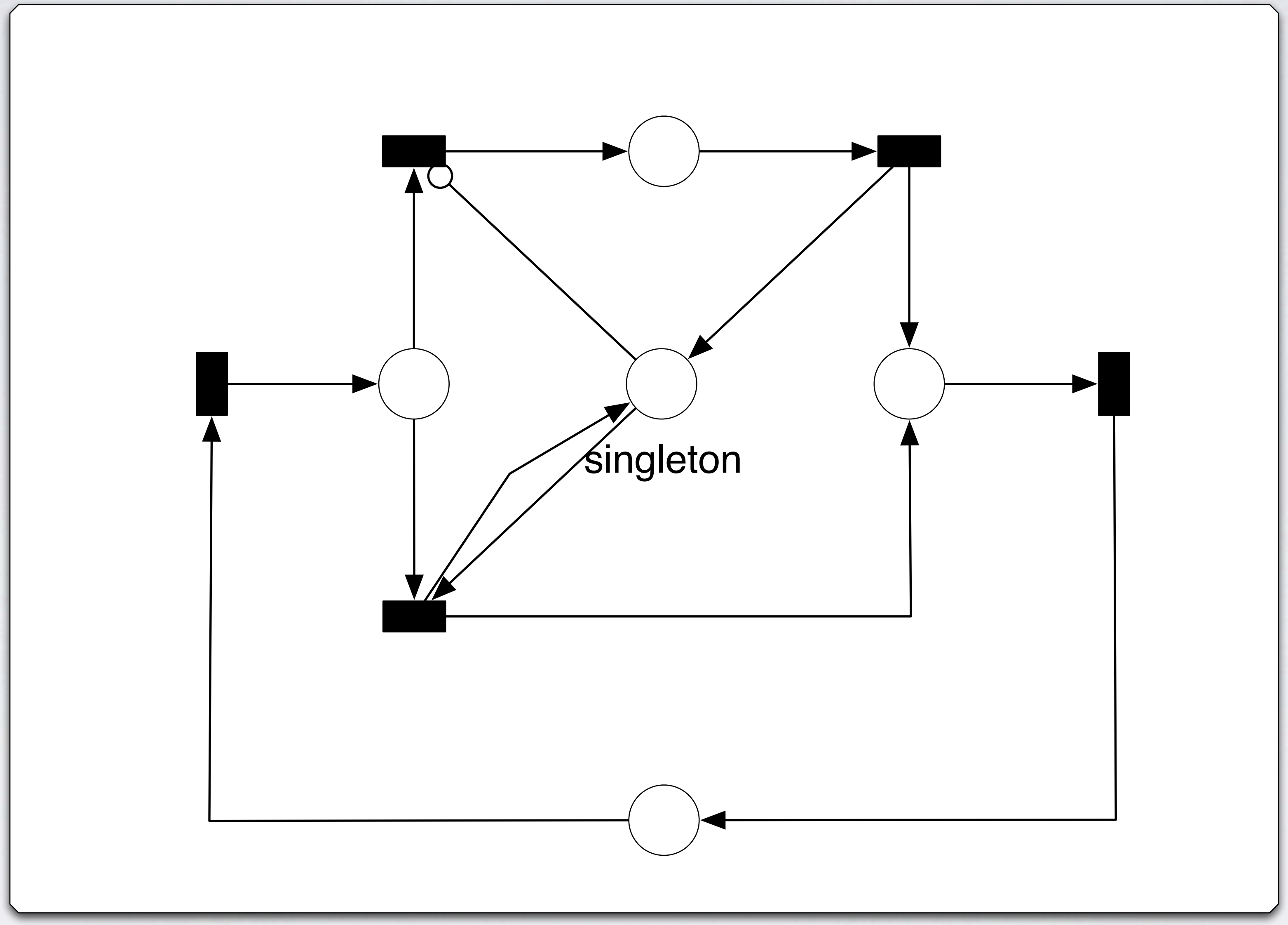
It just works!

... or not

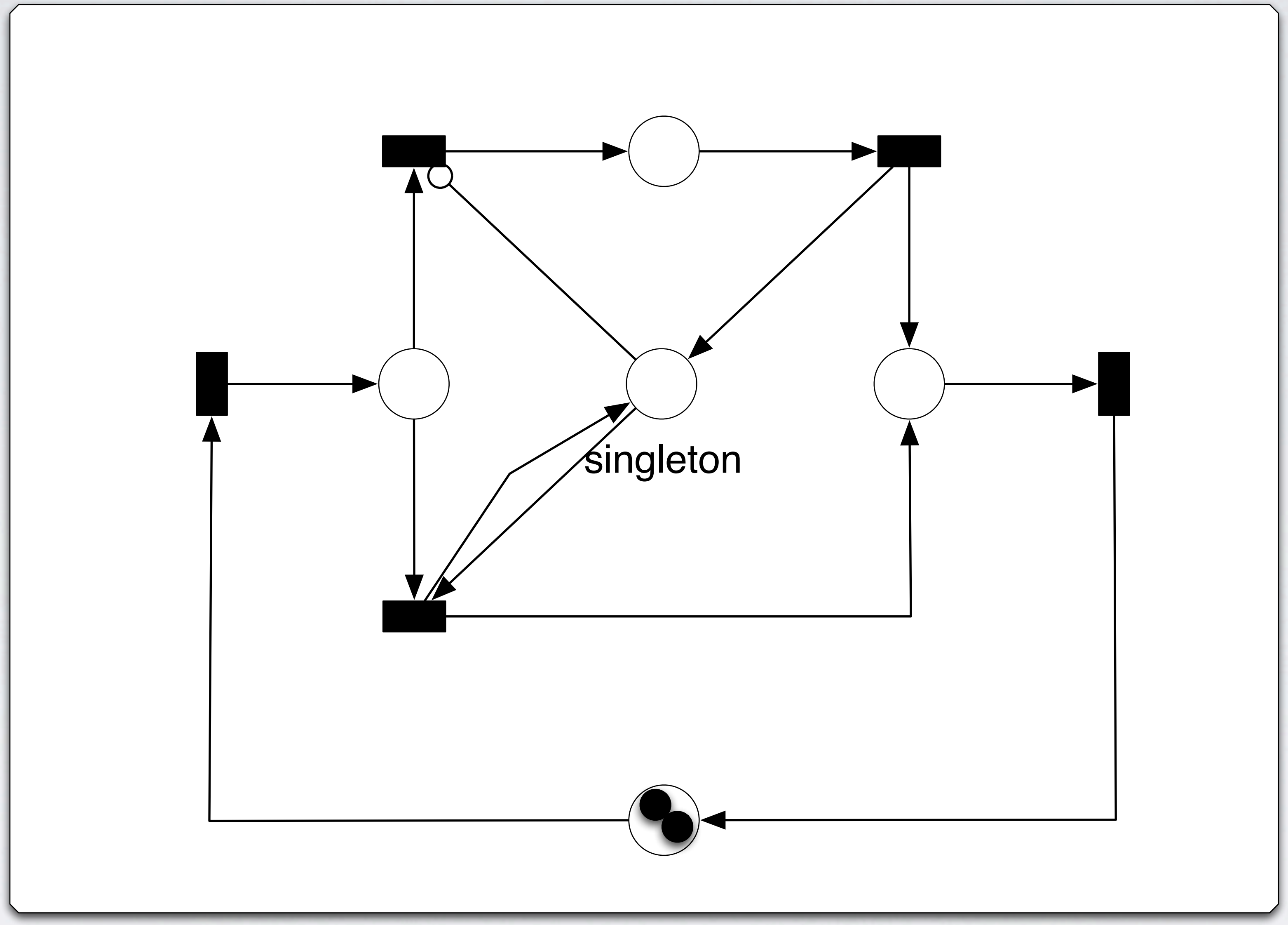
Multithreading



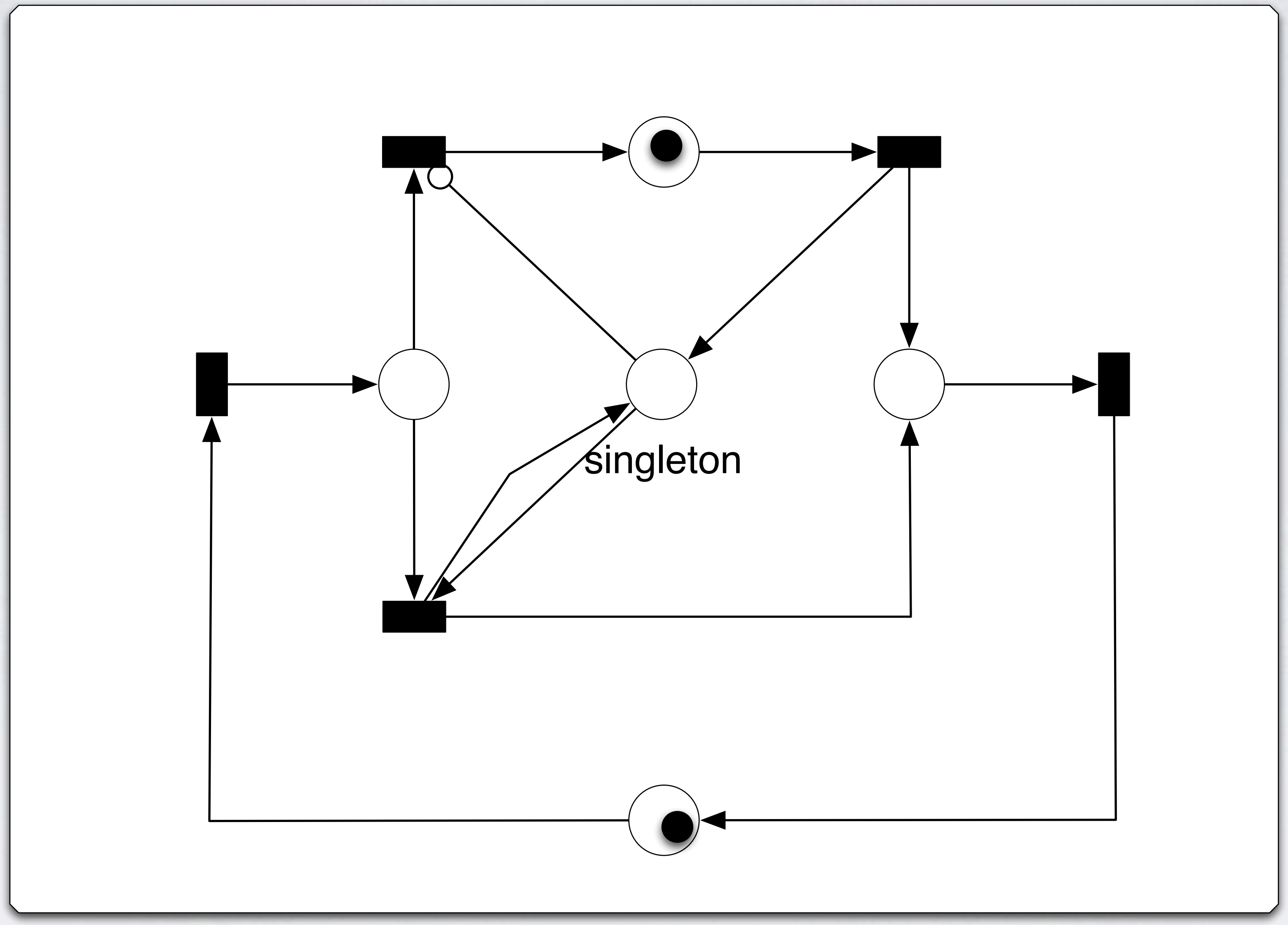
Multithreading



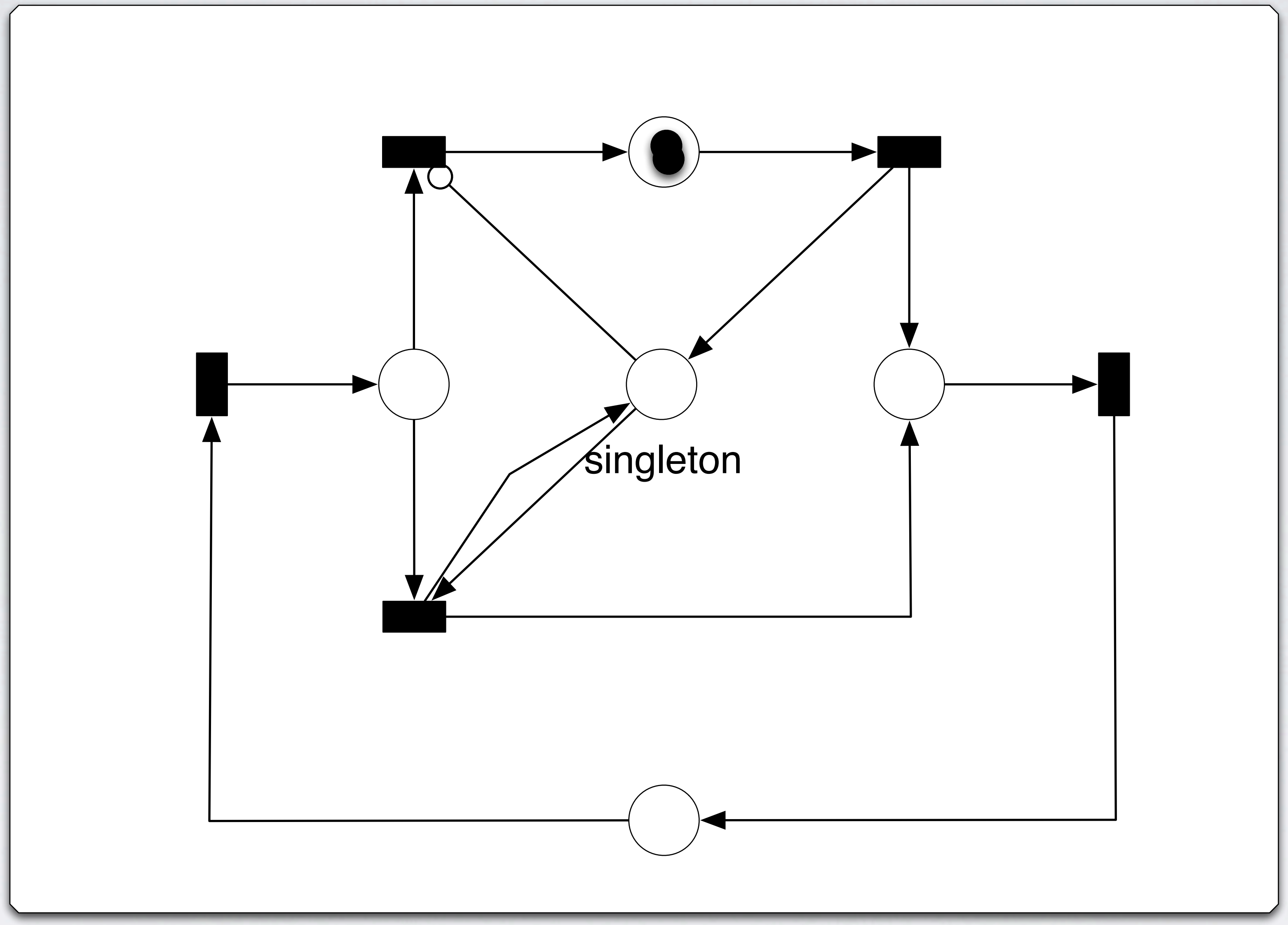
Multithreading



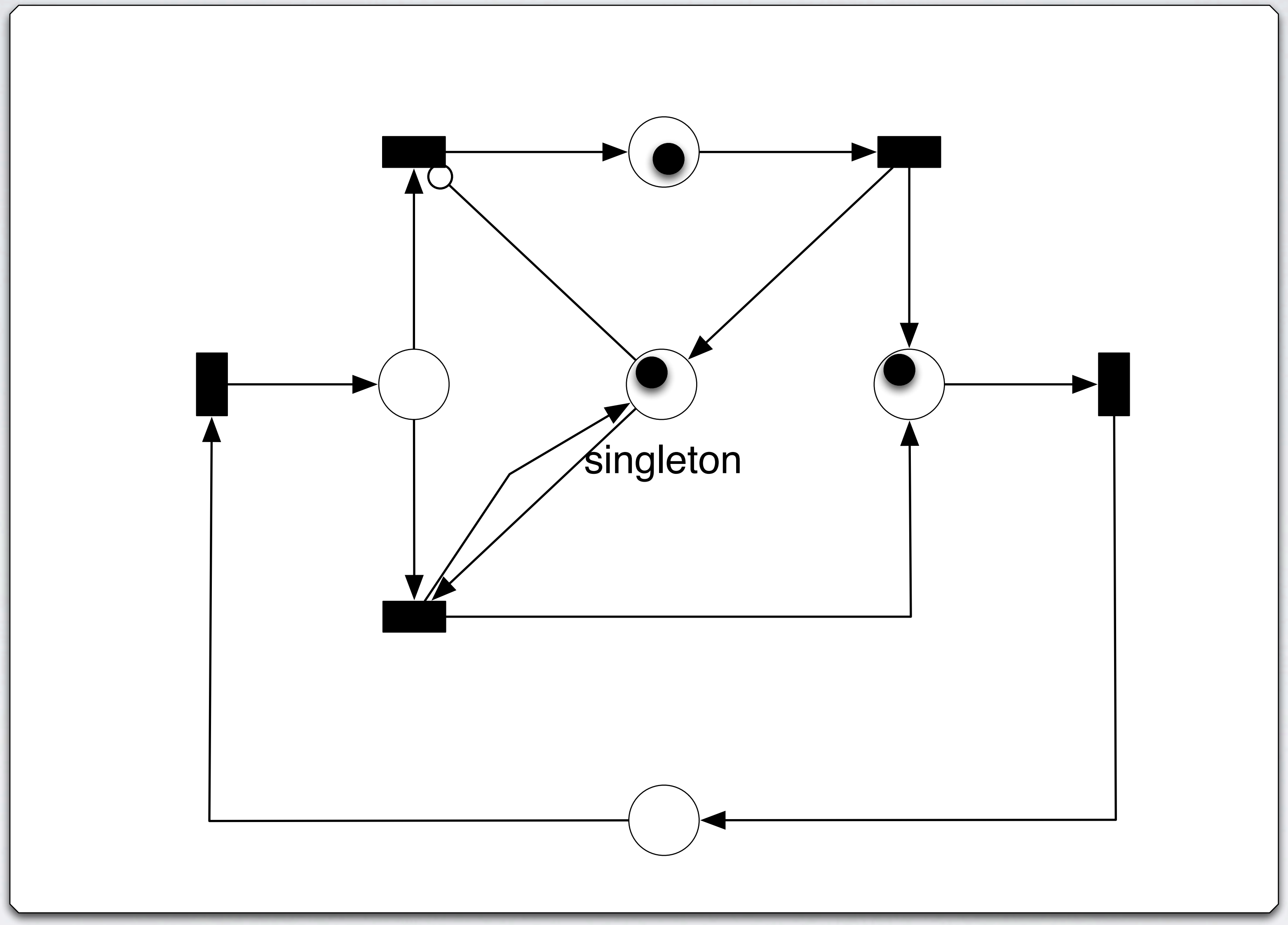
Multithreading



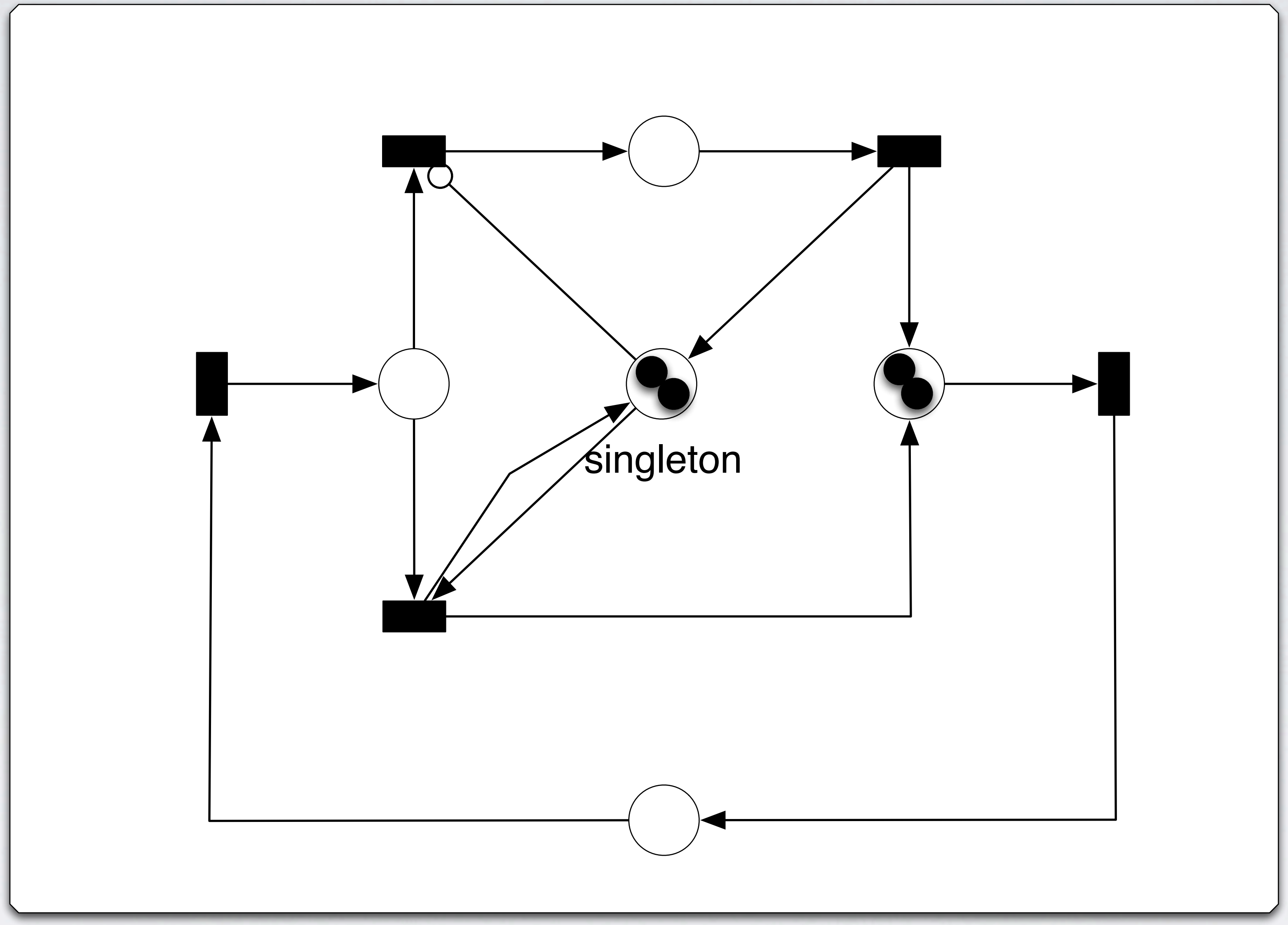
Multithreading



Multithreading



Multithreading



A good solution

```java
class Singleton {
    private static Singleton instance = null;

    private Singleton() {};

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

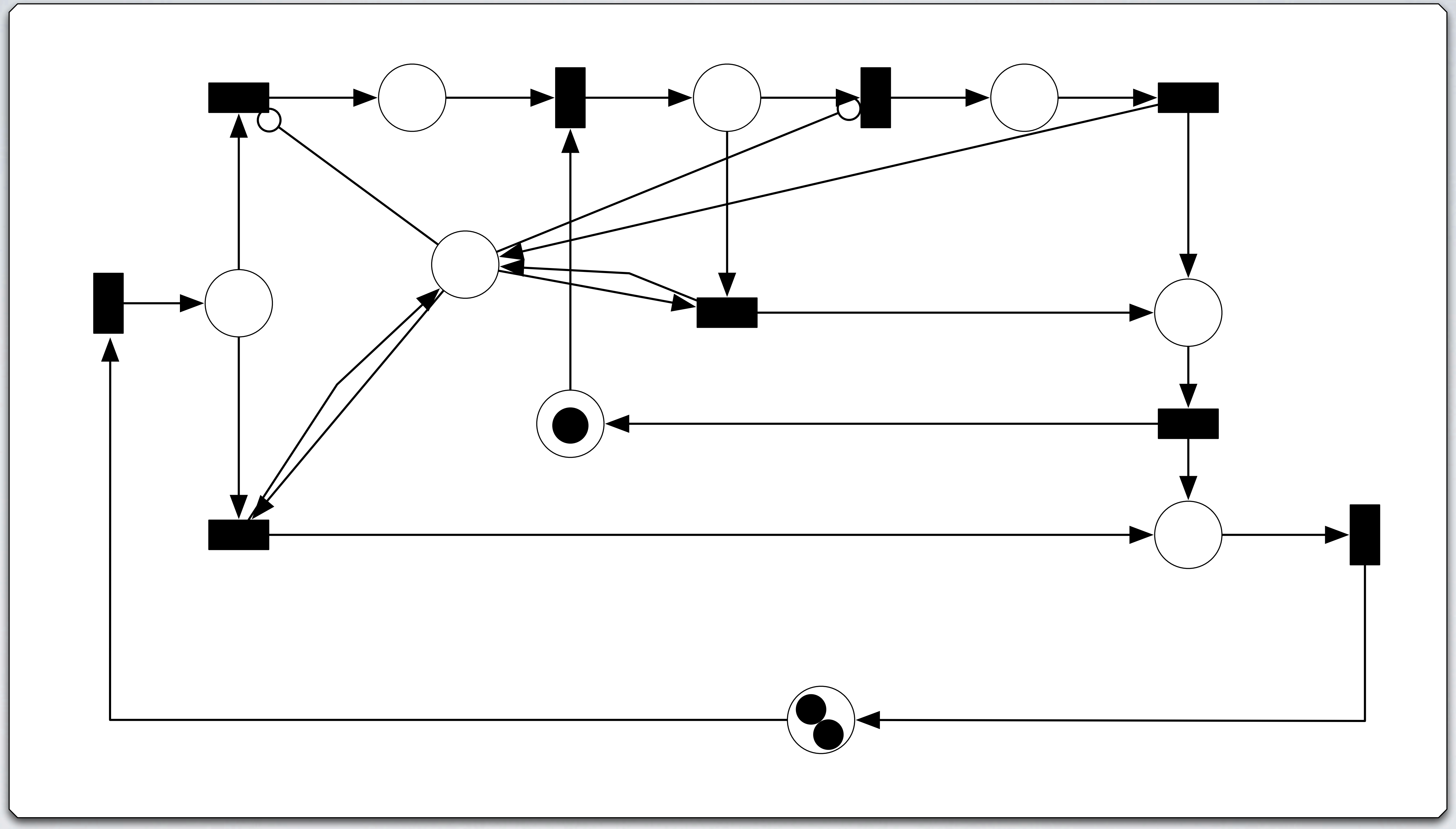
A good solution

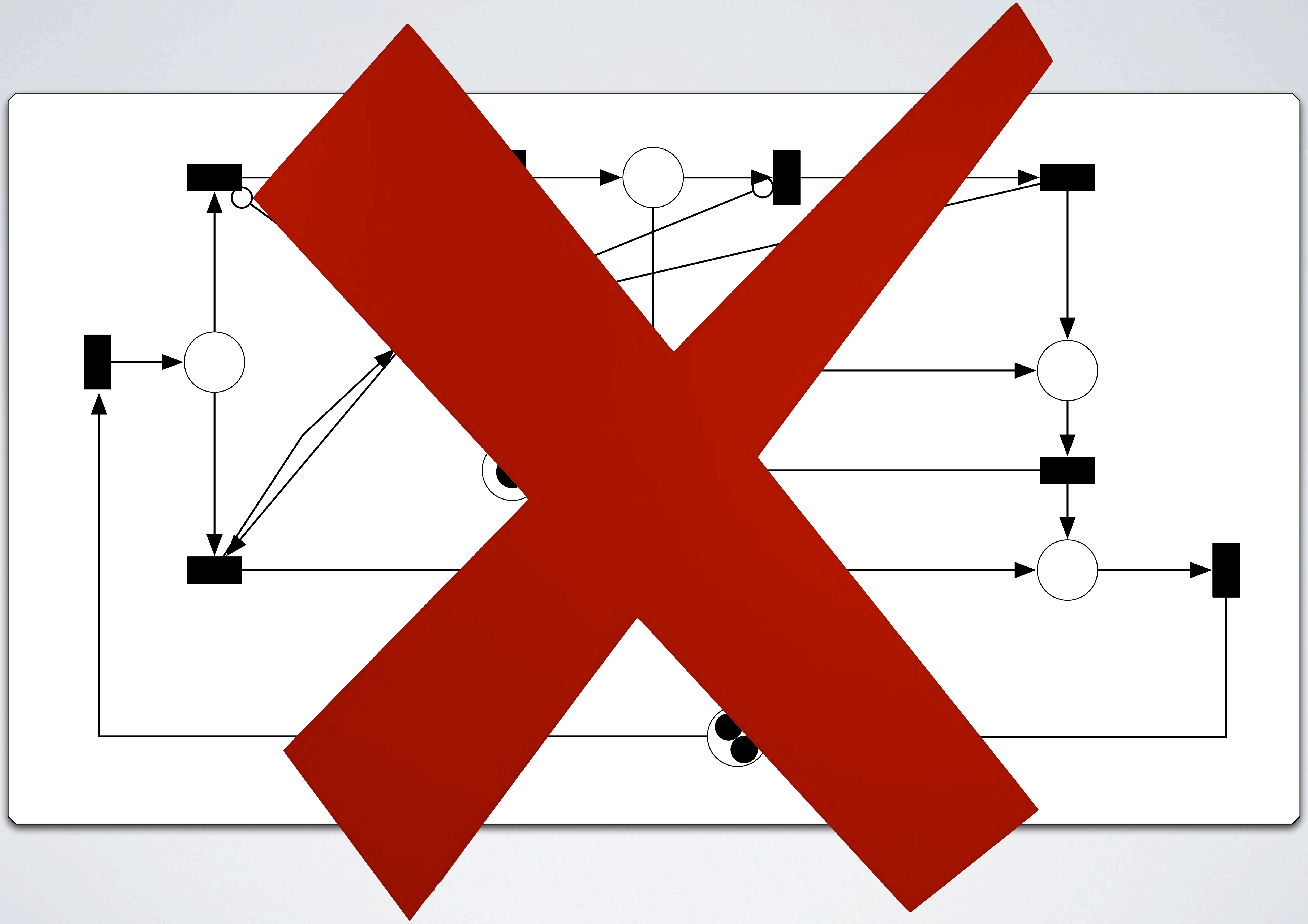
```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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

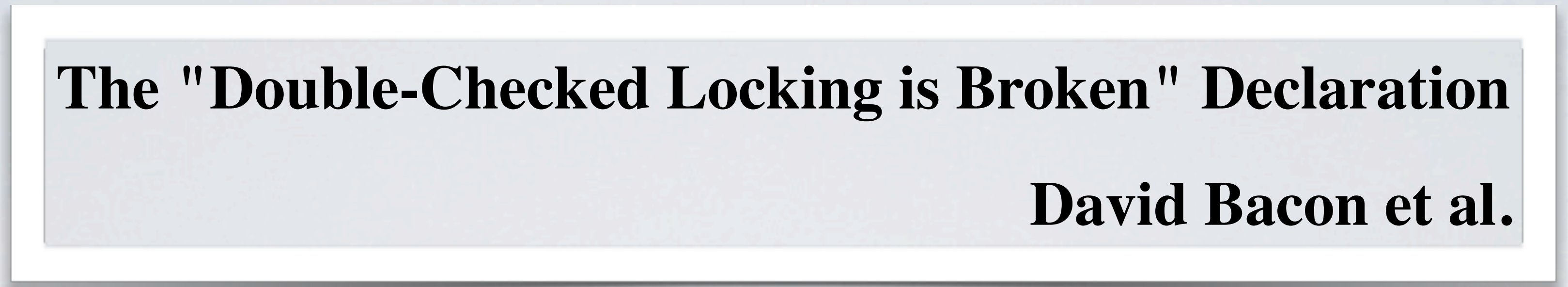
But it is too expensive!

A well-known cheap solution

```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

   public static Singleton getInstance() {
      if (instance == null) {
        synchronized(this) {
          if (instance == null) {
            instance = new Singleton();
          }
        }
      }
      return instance;
   }
}
```





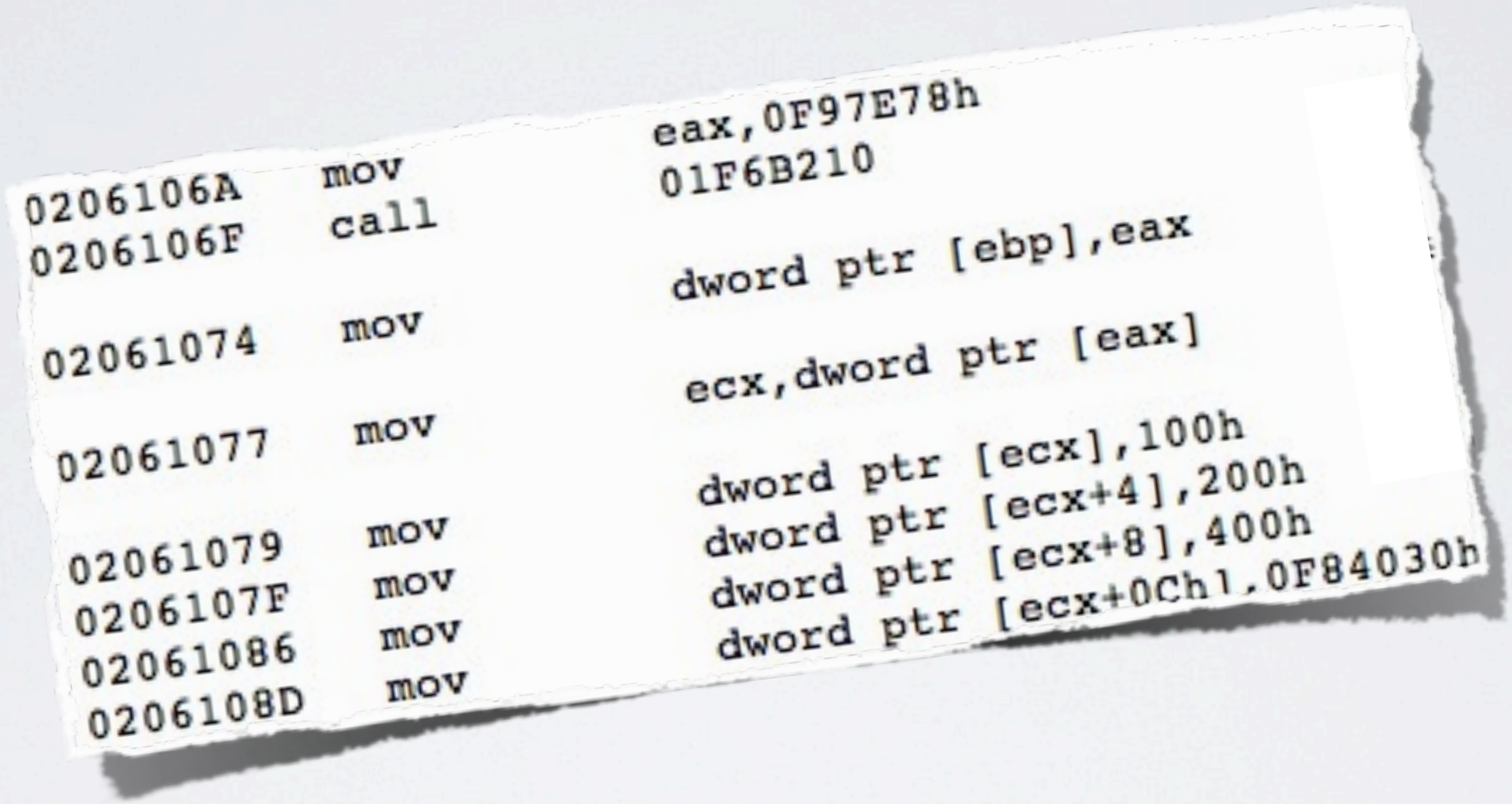


The "Double-Checked Locking is Broken" Declaration

David Bacon et al.

The "Double-Checked Locking is Broken" Declaration

David Bacon et al.



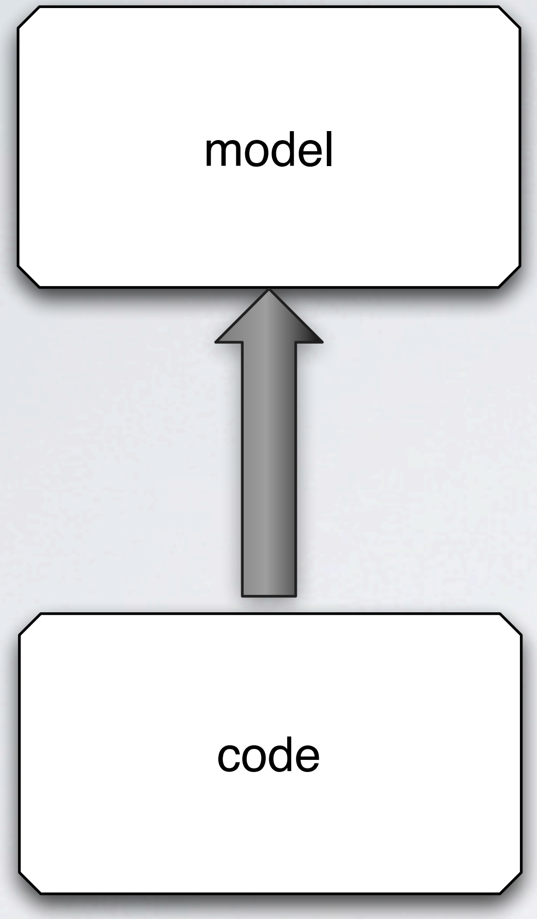
```
0206106A   mov     eax,0F97E78h
0206106F   call    01F6B210
02061074   mov     dword ptr [ebp],eax
02061077   mov     ecx,dword ptr [eax]
02061079   mov     dword ptr [ecx],100h
0206107F   mov     dword ptr [ecx+4],200h
02061086   mov     dword ptr [ecx+8],400h
0206108D   mov     dword ptr [ecx+0Ch],0F84030h
```

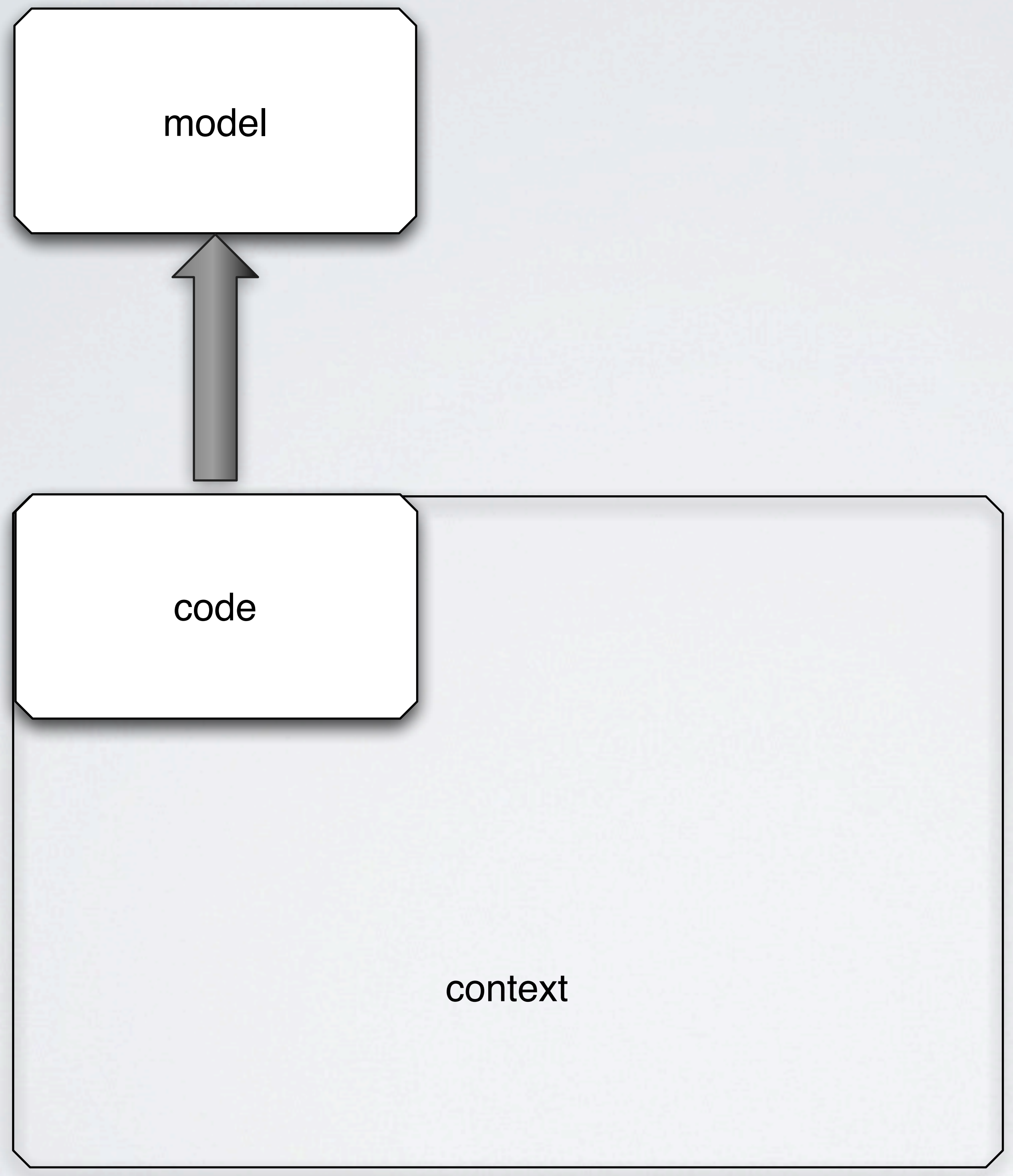
Do we modify the model...

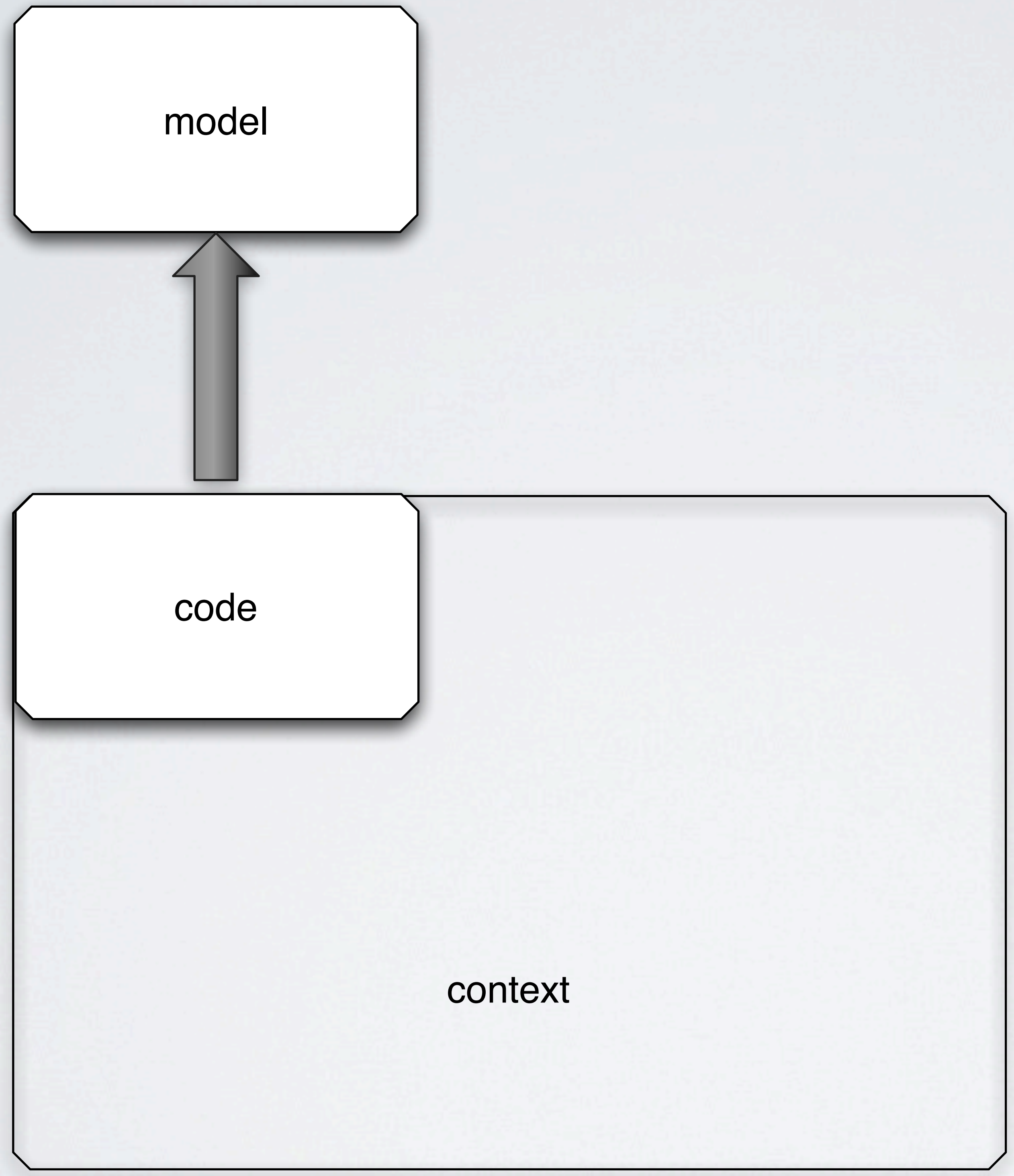
Do we modify the model... again?

Do we modify the model... again?

The transformation is abstracting the bugs!


model
code


model
code
context



Language

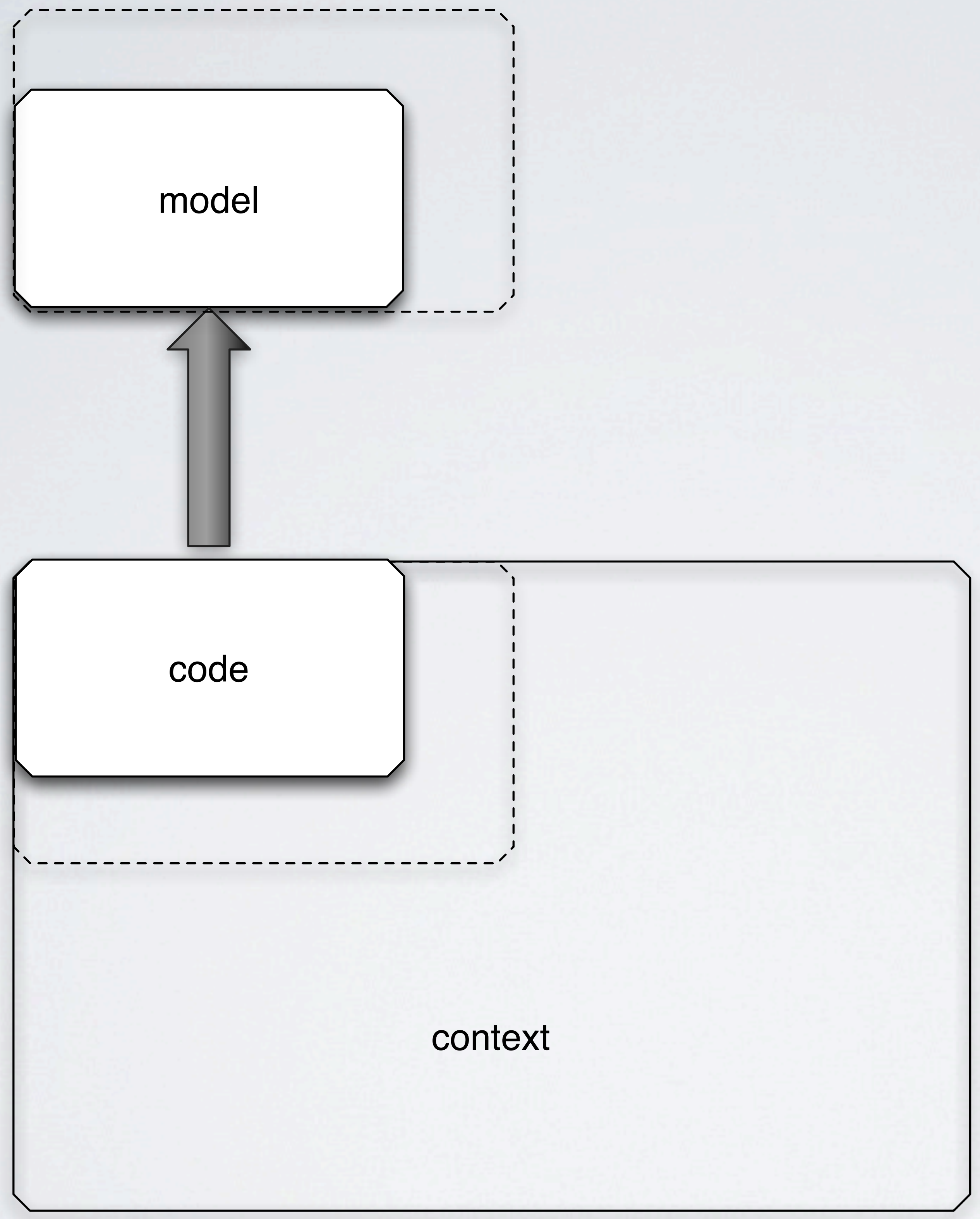
VM / Platform

OS

Hardware

Cosmic rays

...



Language

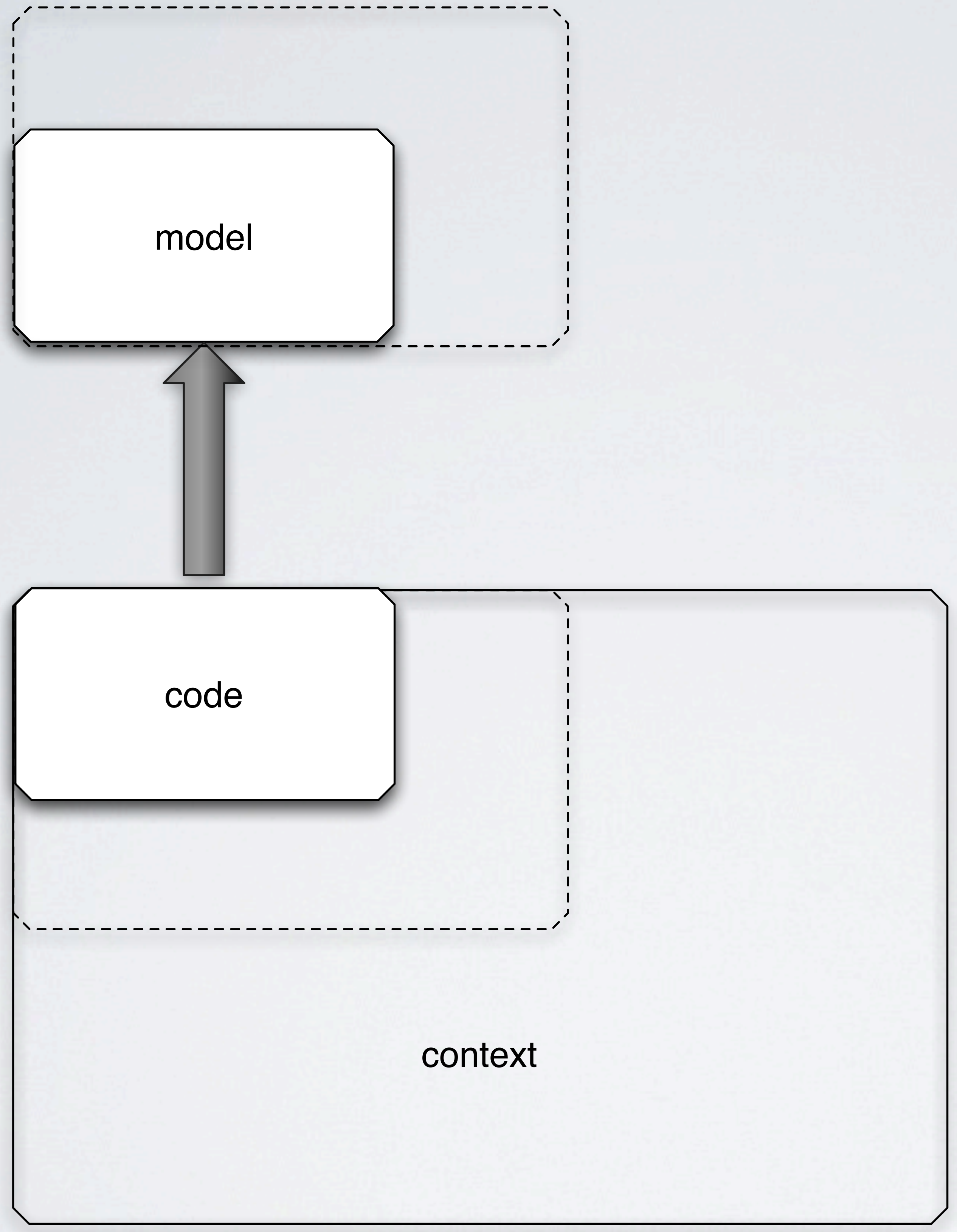
VM / Platform

OS

Hardware

Cosmic rays

...



Language

VM / Platform

OS

Hardware

Cosmic rays

...

“Relevant correctness requirements”

G. Holzmann

“Relevant correctness requirements”

G. Holzmann

G. Candea

“In-vivo and in-vitro executions”

Fool me once, shame on you;
fool me twice, shame on me

```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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

```
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

   public static Singleton getInstance() {
      if (instance == null) {
        synchronized(this) {
          if (instance == null) {
            instance = new Singleton();
          }
        }
      }
      return instance;
   }
}
```

```java
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

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

Multithreading

```java
class Singleton {
   private static Singleton instance = null;

   private Singleton() {};

   public static Singleton getInstance() {
      if (instance == null) {
        synchronized(this) {
          if (instance == null) {
            instance = new Singleton();
          }
        }
      }
      return instance;
   }
}
```

Processor optimisations

Risk involved	Program	Expected behavior
Multithreading	☕	Multiple singletons
Multithreading	☕	Delays
Execution in Windows	☕	OS crash
...	...	...

Risk involved	Program	Expected behavior
Multithreading	☕	Multiple singletons
Multithreading	☕	Delays
Execution in Windows	☕	OS crash
...	...	...

Witnesses database

A. Podelski

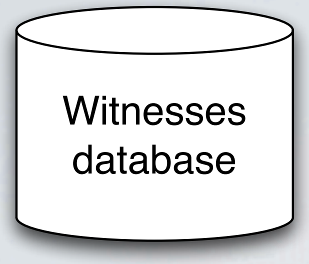
A. Podelski

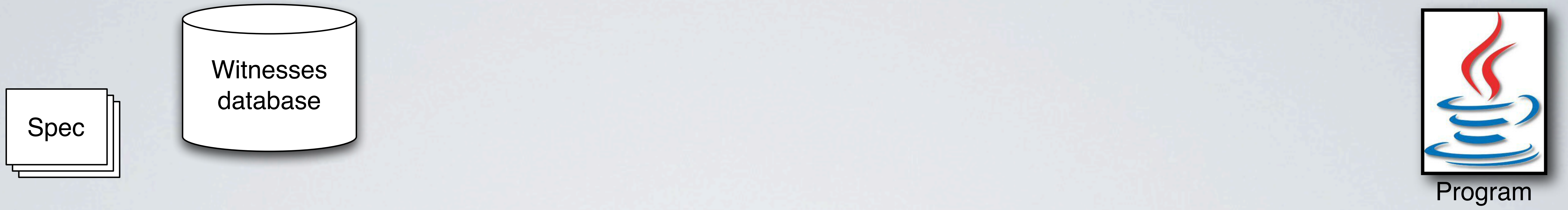
“Recycling of Program Correctness Proofs”

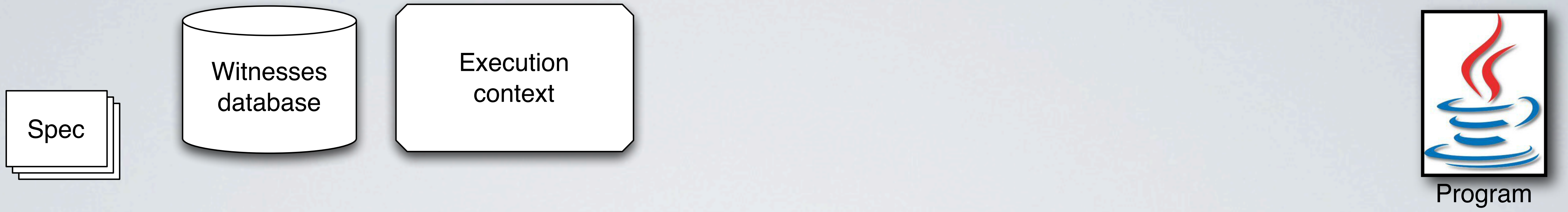
A. Podelski

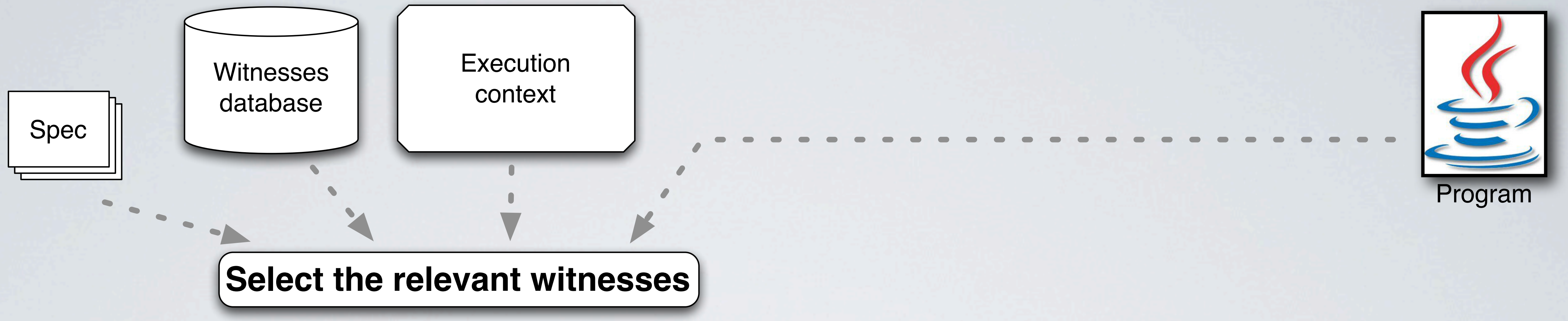
“Recycling of Program Correctness Proofs”

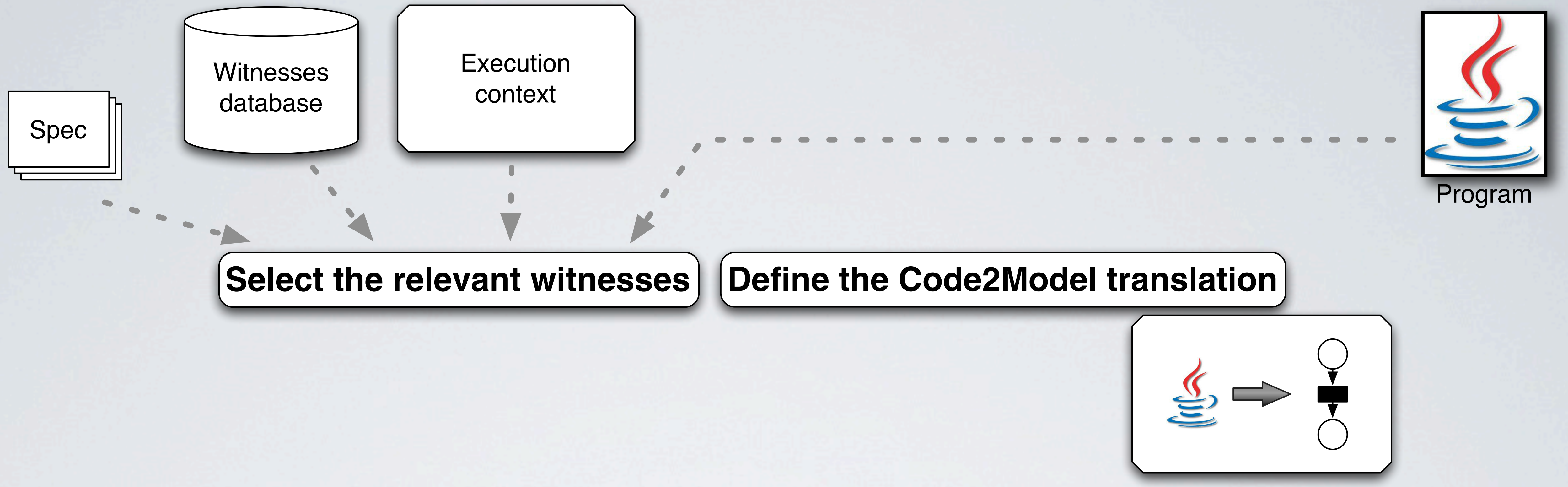
“A crazy idea”

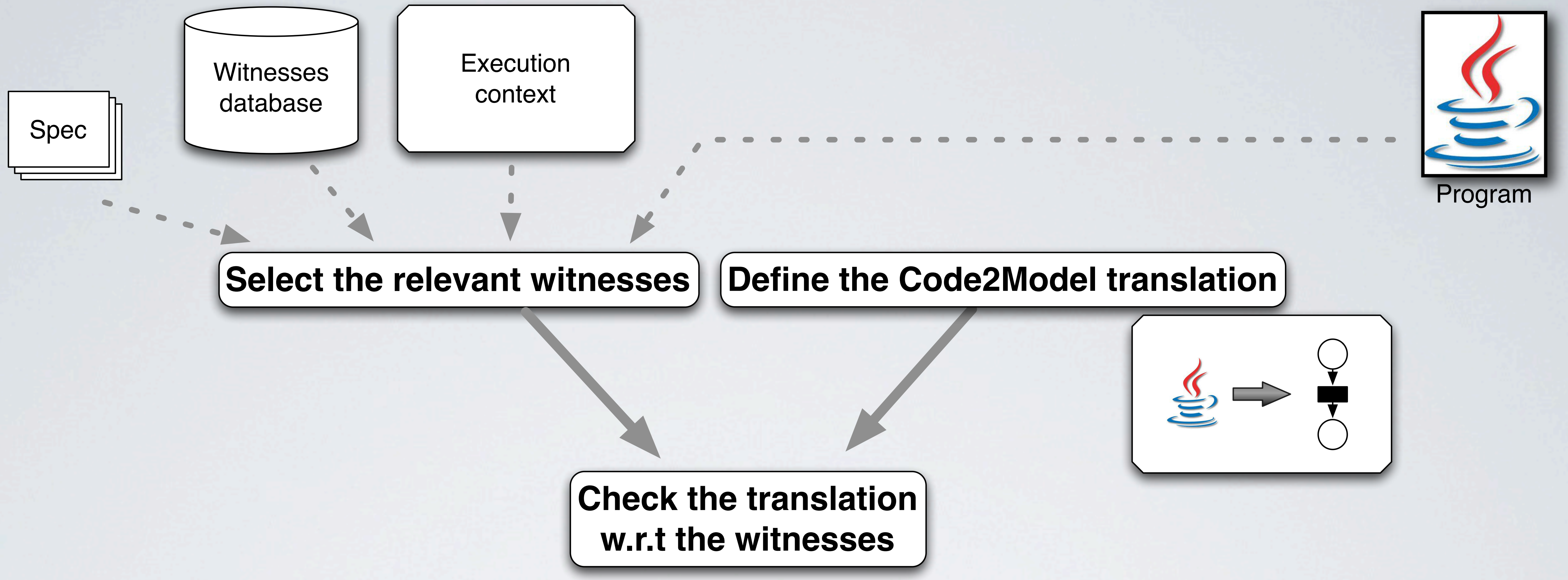

Witnesses database

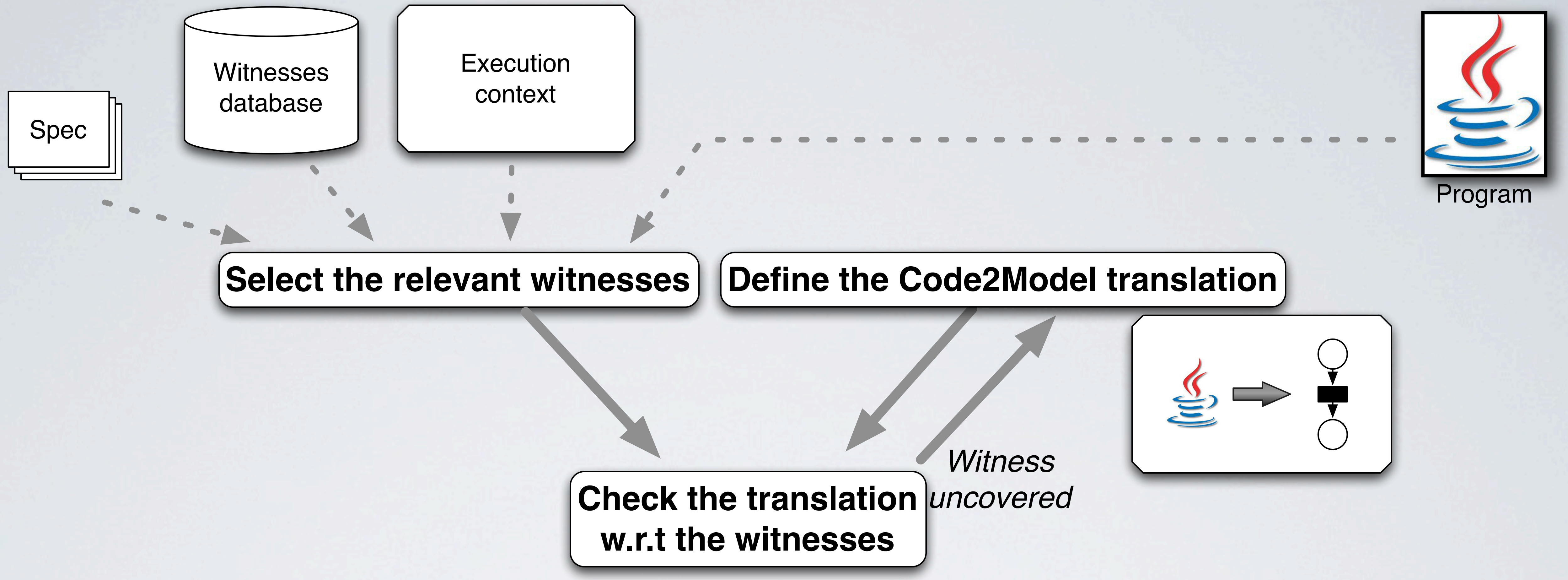

Spec
Witnesses database
Program

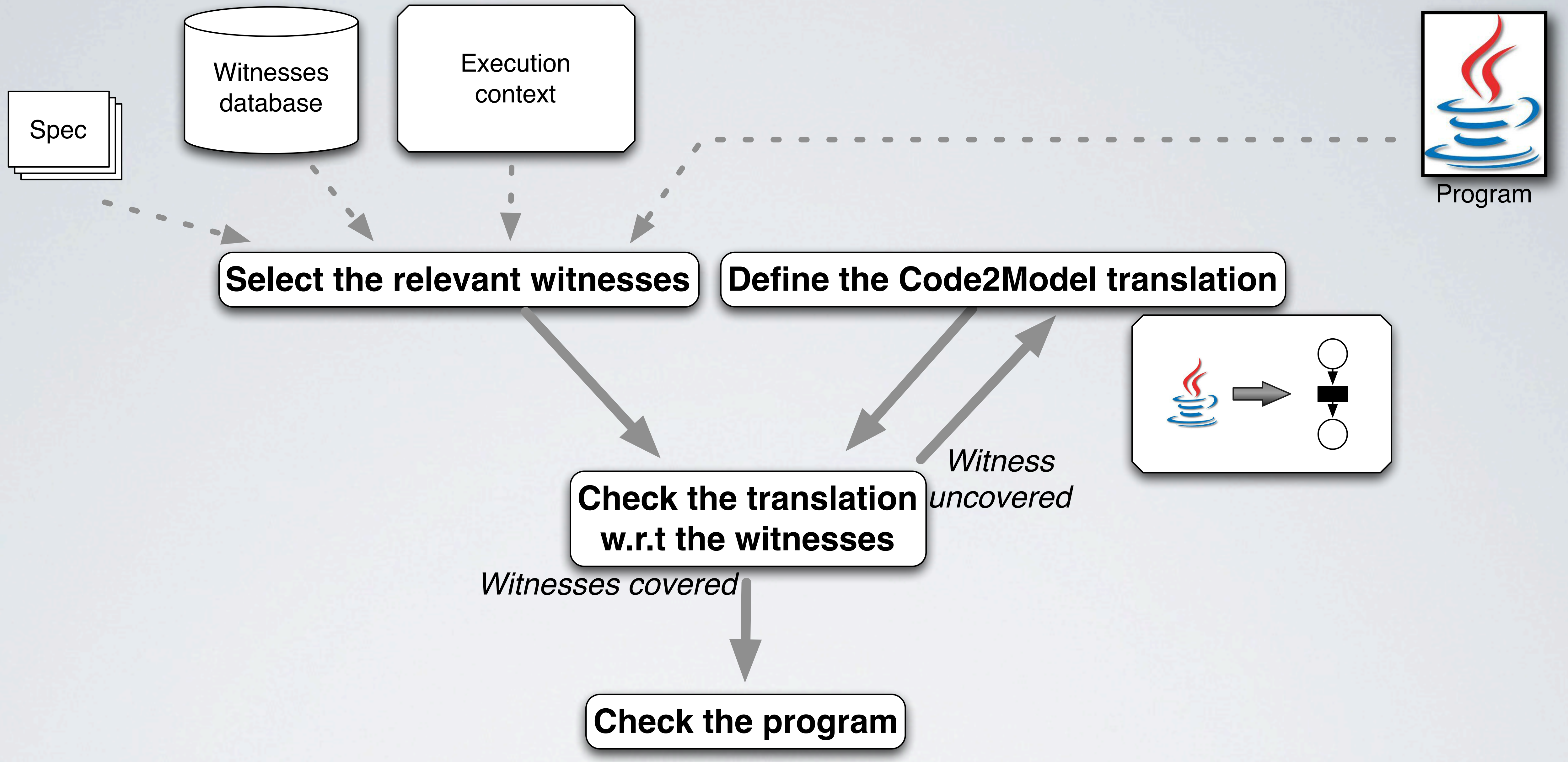

Spec
Witnesses database
Execution context
Program

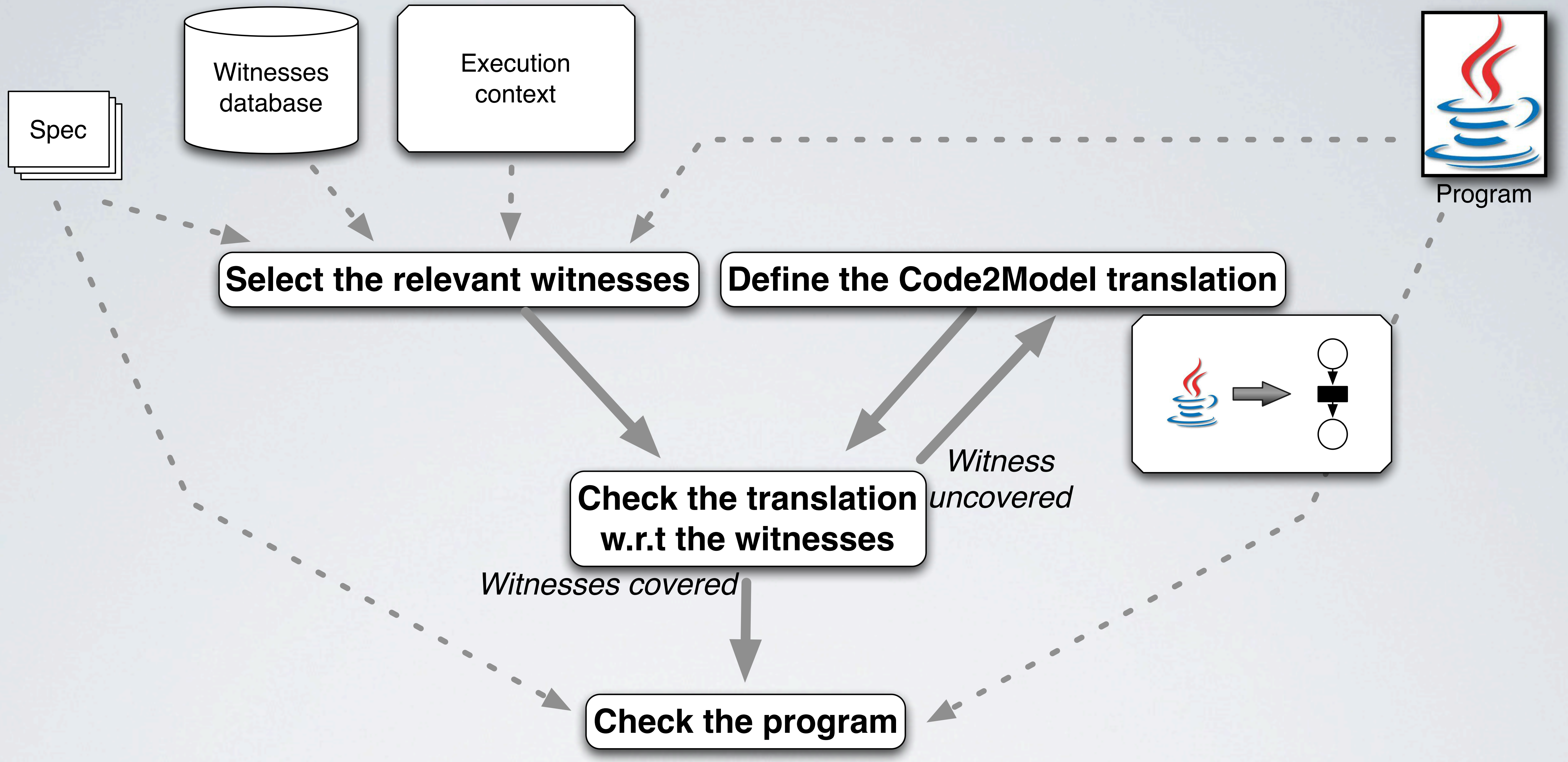

Spec
Witnesses database
Execution context
Program
Select the relevant witnesses

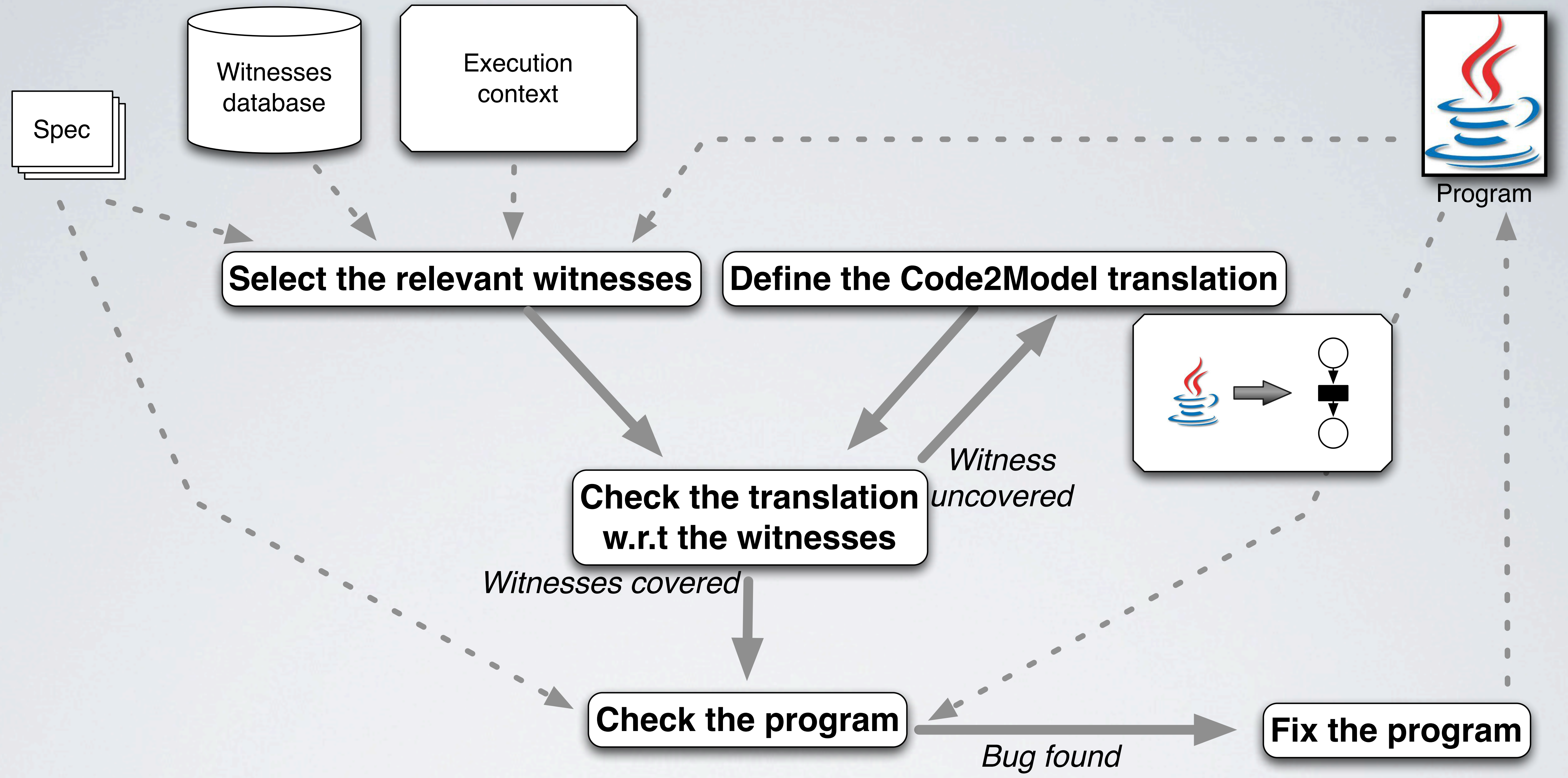

Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation

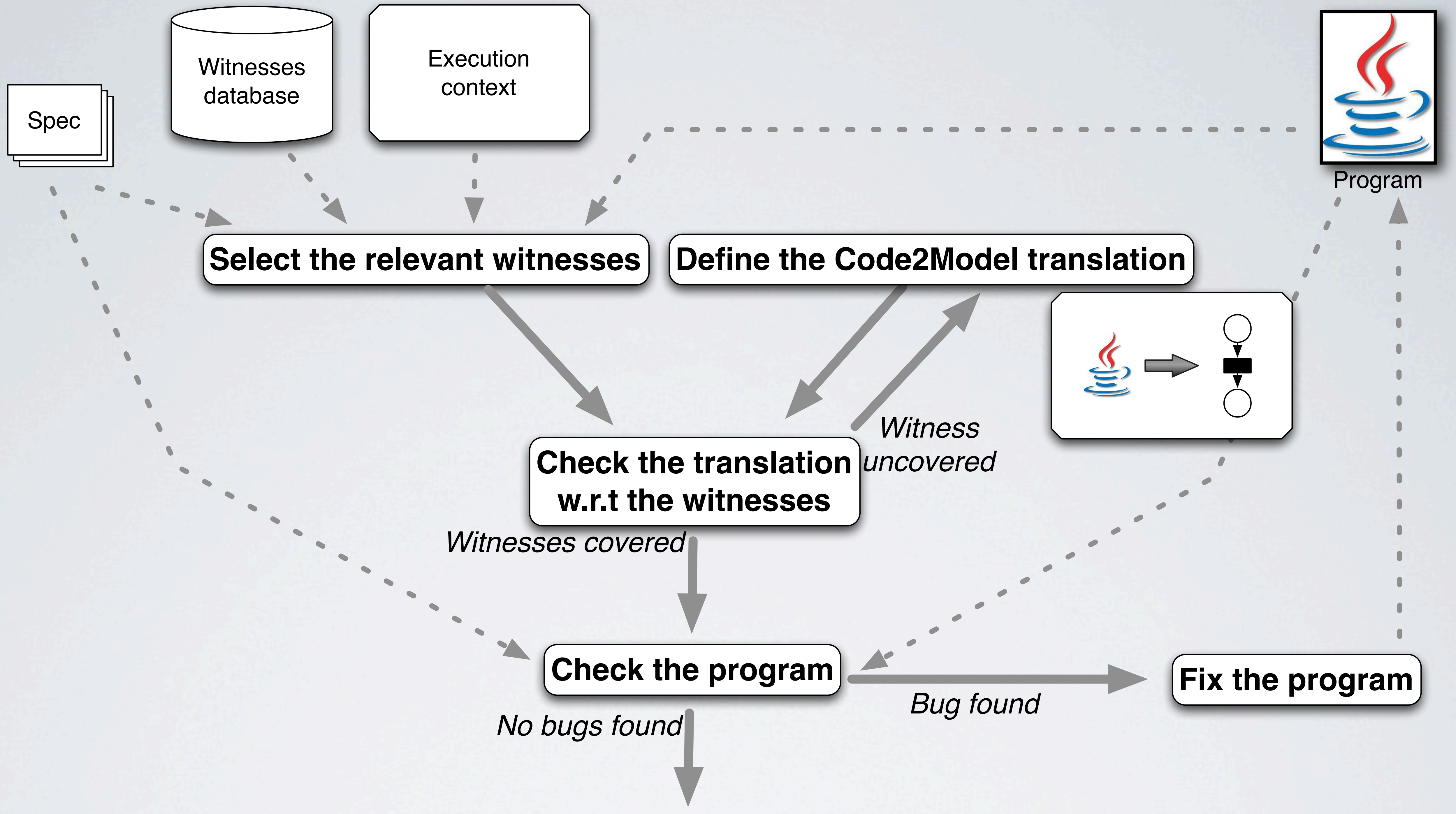

Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation
Check the translation w.r.t the witnesses

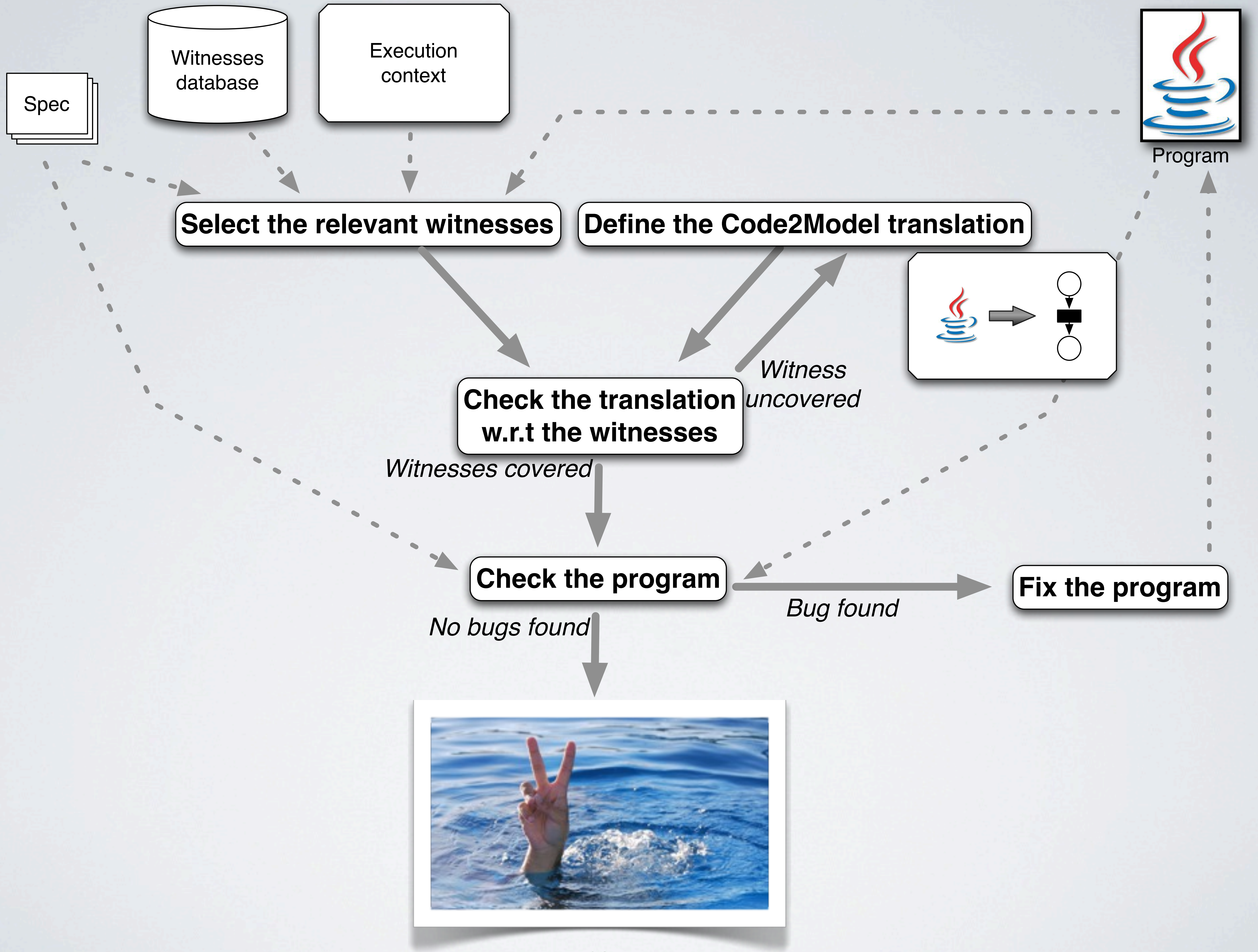

Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation
Check the translation w.r.t the witnesses
Witness uncovered


Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation
Witness uncovered
Check the translation w.r.t the witnesses
Witnesses covered
Check the program


Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation
Witness uncovered
Check the translation w.r.t the witnesses
Witnesses covered
Check the program


Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation
Witness uncovered
Check the translation w.r.t the witnesses
Witnesses covered
Check the program
Bug found
Fix the program


Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation
Witness uncovered
Check the translation w.r.t the witnesses
Witnesses covered
Check the program
Bug found
Fix the program
No bugs found



Spec
Witnesses database
Execution context
Program
Select the relevant witnesses
Define the Code2Model translation
Check the translation w.r.t the witnesses
Witness uncovered
Witnesses covered
Check the program
Bug found
Fix the program
No bugs found

Experience-based model refinement

Not automatic!

Experience-based model refinement

Thank you!