Java PathFinder

Nastaran Shafiei
nastaran@cse.yorku.ca

York University, Toronto

Outline

- Why software verification?
- Why model checking?
- What is Java PathFinder (JPF)?
- Major components of JPF
  - JVM
  - Search
- Listeners
- Examples

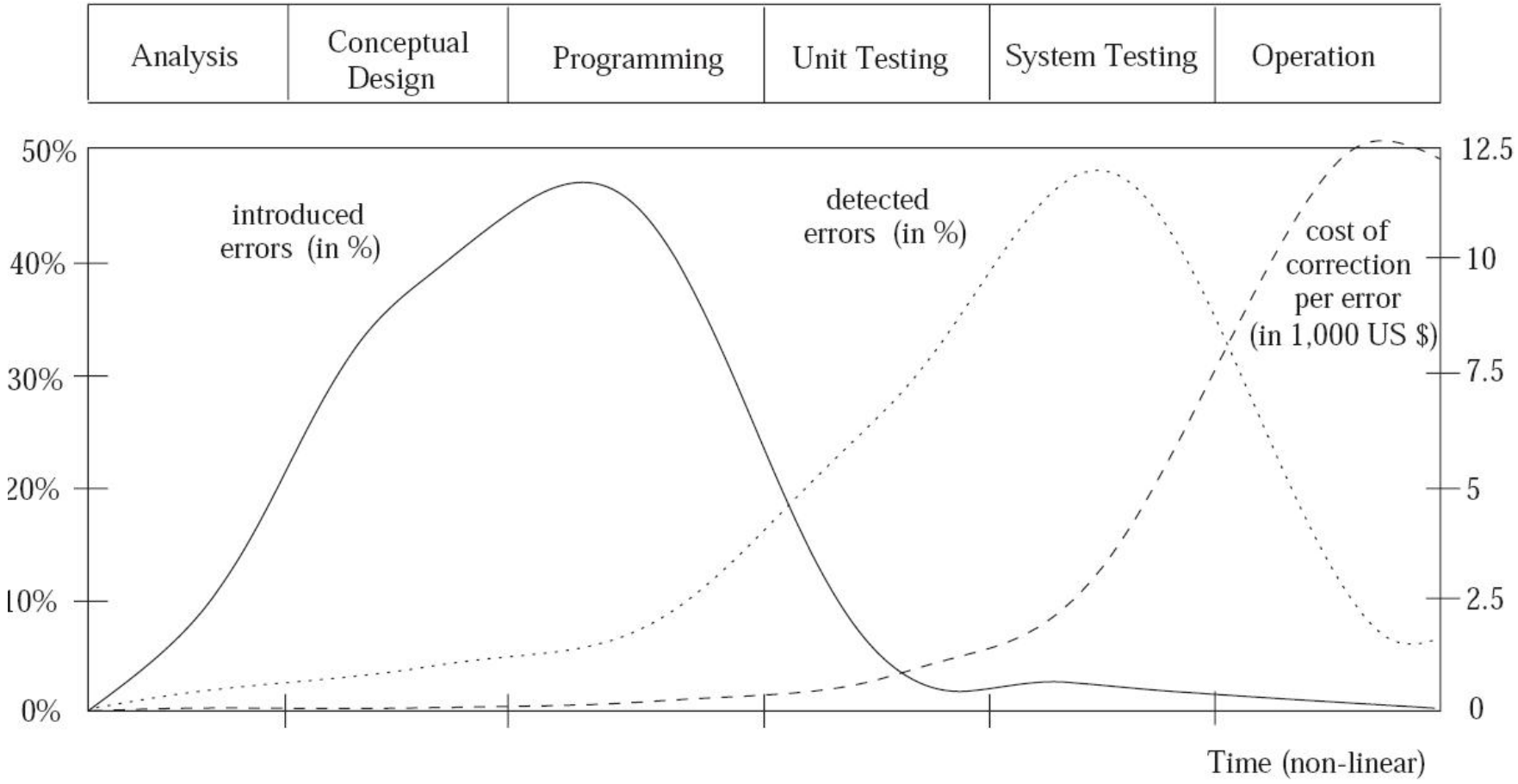
Software Verification

- Involvement of software in human life
- Unreliability
  - Paul A. Strassmann, former CIO of Xerox :*software can easily rate among the most poorly constructed, unreliable and least maintainable technological artifacts ever invented by man*.
- Costly Bugs
  - Intel Pentium FDIV bug of 1994
    - Reason: error in floating point divisions
    - Cost: $500 million

Software Verification

- Costly Bugs
  - Year 2000
    - Reason: representing a 4 digit year by 2 digits
    - Cost: $500 billion
  - Northeast Blackout of 2003
    - Reason: race condition
    - Cost: between $7 and $10 billion
  - According to NIST in 2002 software bugs cost the U.S. economy about $59.5 billion annually which is about 0.6 percent of the gross domestic product

Software lifecycle



- Sofware lifecycle and error introduction, detection, and repair costs

Different Verification Techniques

- Testing
  - Applied to all types of softwares ✓
  - Cost-effective ✓
  - Impossible to capture all the potential executions ✗
  - Non-automatic ✗
- Theorem Proving
  - Applied to infinite state systems ✓
  - Used by an expert ✗
  - Time-consuming ✗

Different Verification Techniques

- Model checking
  - Examines all possible executions ✓
  - Automatic ✓
  - Restricted to finite state systems ✗

  - Different categories:
    - Applied on system model
    - Applied on actual system

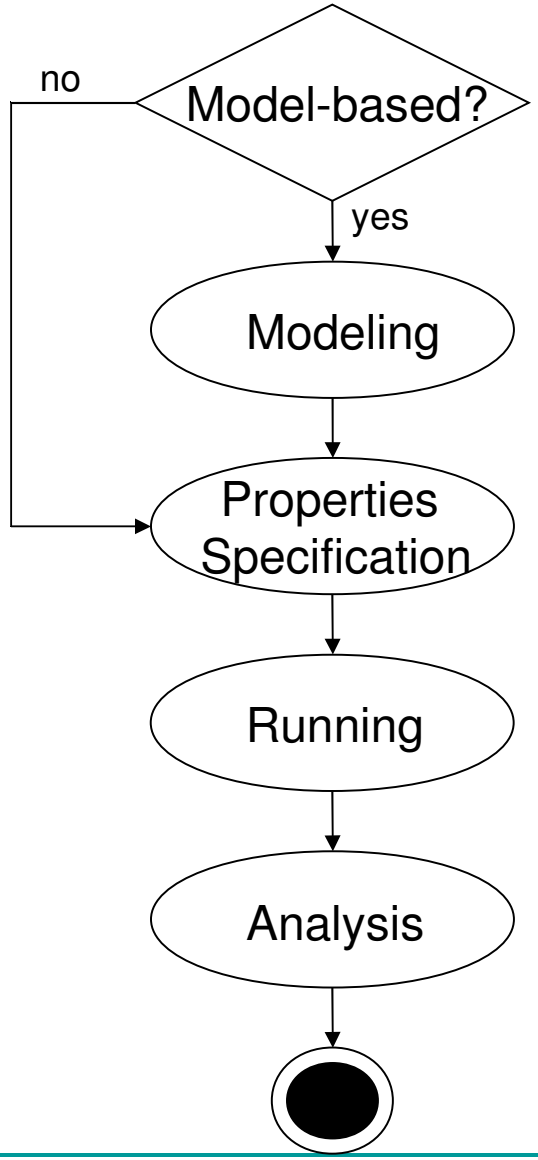
Model Checkers

1. Model-based:

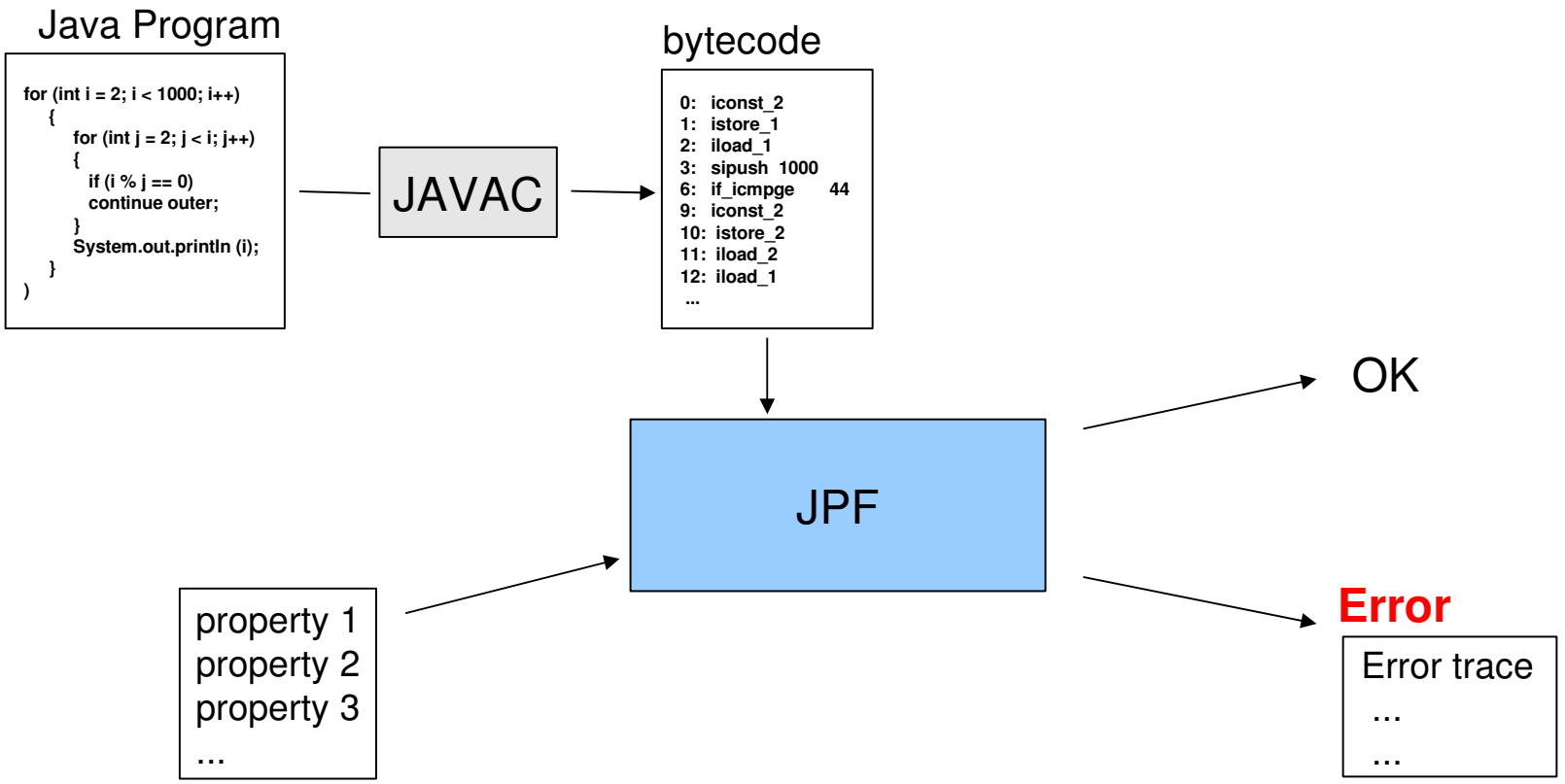
   Spin, SMV, ...

1. System-based:

   BLAST, SLAM, Bandera, JPF, ...



Java PathFinder (JPF)

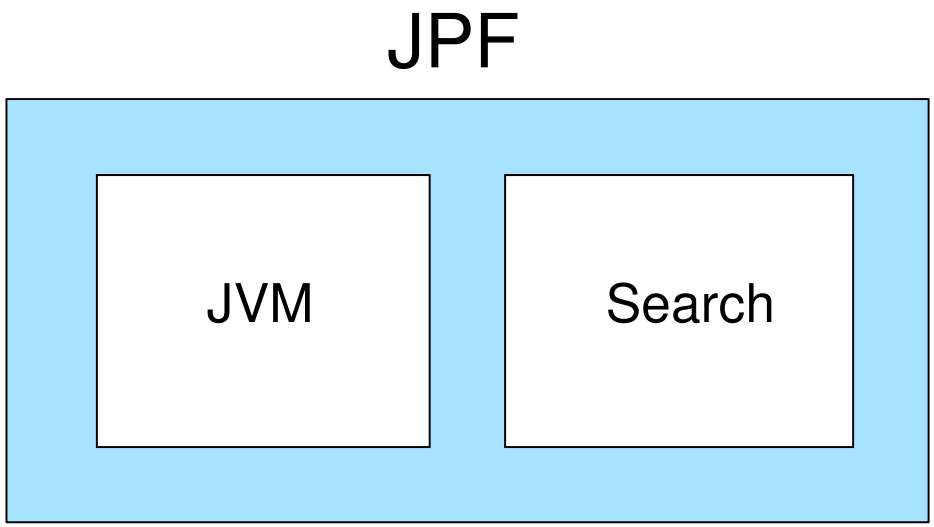


History of JPF

- 1999: Started at NASA
  - Translator from Java to Promela
- 2000: Re-factored as JVM, Why?
  - Promela cannot represents all features of Java.
  - Java source code is not always available.
- 2005: Open sourced on Sourceforge
- 2009: Move to their own server, Mercurial
  - http://babelfish.arc.nasa.gov/trac/jpf
- Today, is a mature tool, 100s of active users, more than 100 downloads monthly

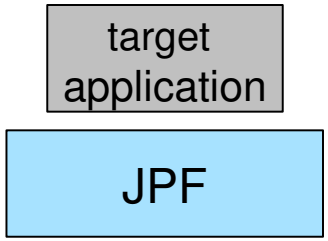
JPF Main Components

JPF consists of two major components



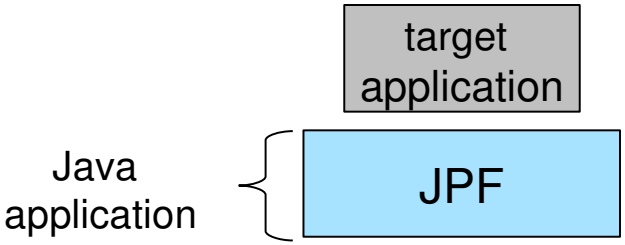
JPF is a JVM

- JPF is a VM for Java bytecodes
  - Executes the Java application
  - Handles all bytecode instructions generated by a Java compiler



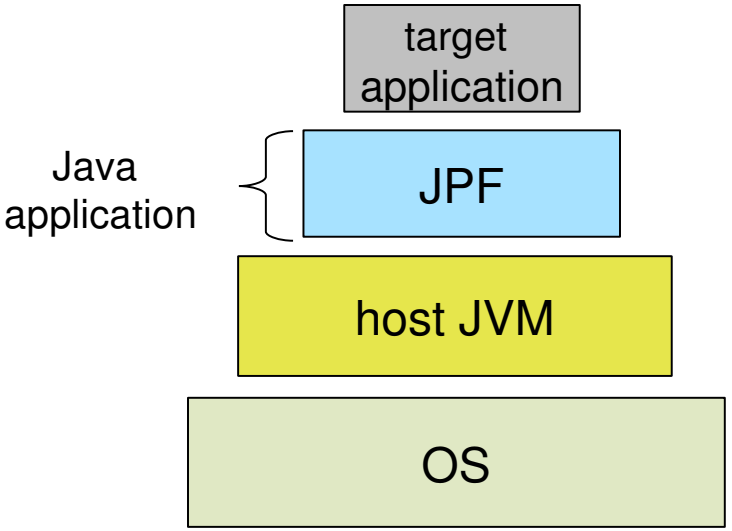
JPF is a JVM

- JPF is a VM for Java bytecodes.
  - Executes the Java application
  - Handles all bytecode instructions generated by a Java compiler
- JPF is written in Java



JPF is a JVM

- JPF is a VM for Java bytecodes
  - Executes the Java application
  - Handles all bytecode instructions generated by a Java compiler
- JPF is written in Java



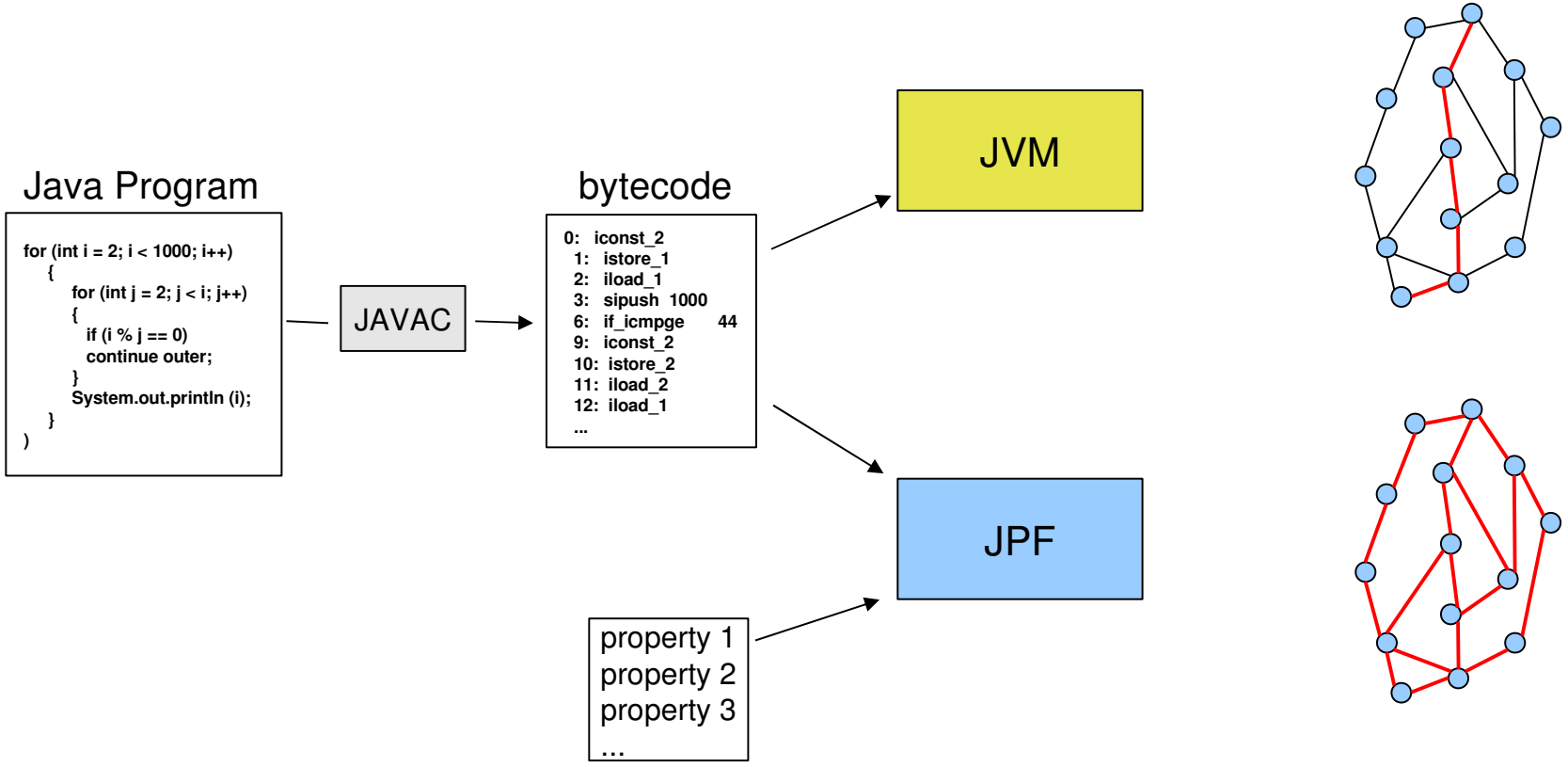
JPF is a Special JVM

1. As executing it checks for certain properties and produces error trace

   - Unhandled exception
     - e.g. NullPointerException, ArrayStoreException, …
   - User-defined assertion
     - assert(boolean-expression)
   - Deadlock
     - i.e. threads mutually wait for each other to progress
   - (Data) race
     - Conflicting access to the a variable

JPF is a Special JVM

2. It goes through all possible executions



JPF is a Special JVM

2. It goes through all possible executions

Example:

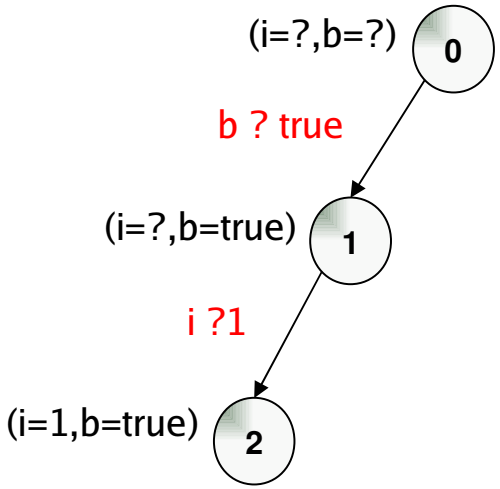
```
int i;
boolean b;

b = (new Random()).nextBoolean();

if(b)
   i = 1;
else
   i = 0;
```

JPF is a Special JVM

2. It goes through all possible executions

Example:

```
int i;
boolean b;

b = (new Random()).nextBoolean();

if(b)
   i = 1;
else
   i = 0;
```

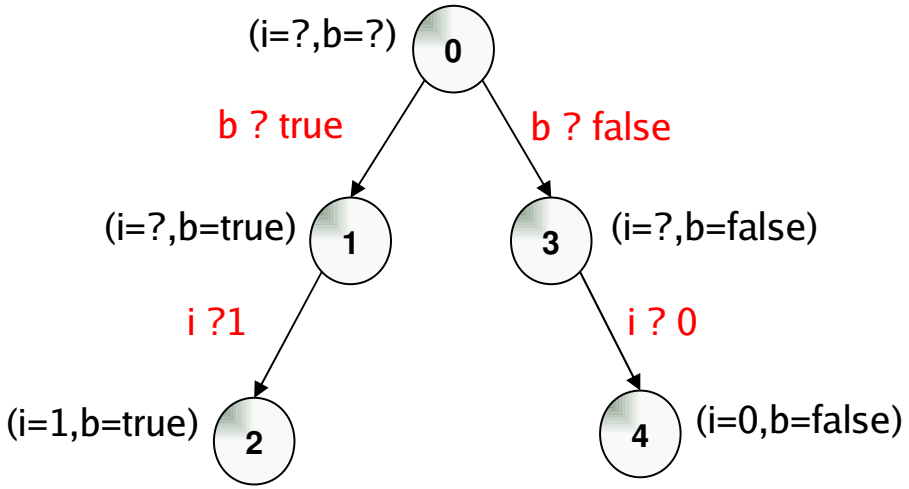


JPF is a Special JVM

2. It goes through all possible executions

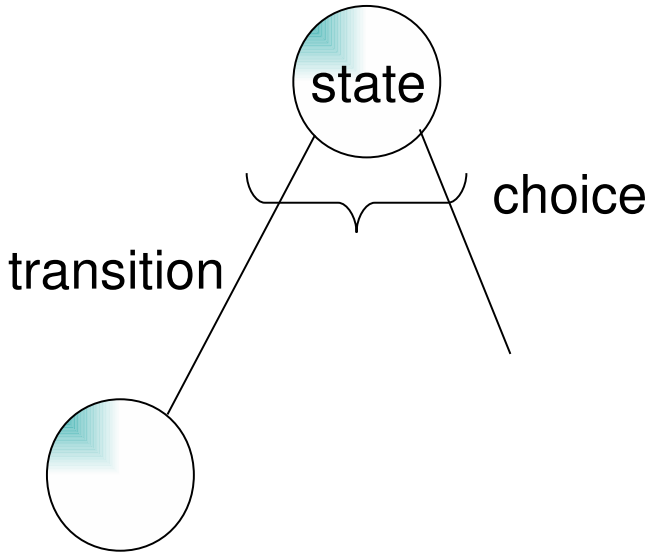
Example:

```
int i;
boolean b;

b = (new Random()).nextBoolean();

if(b)
   i = 1;
else
   i = 0;
```

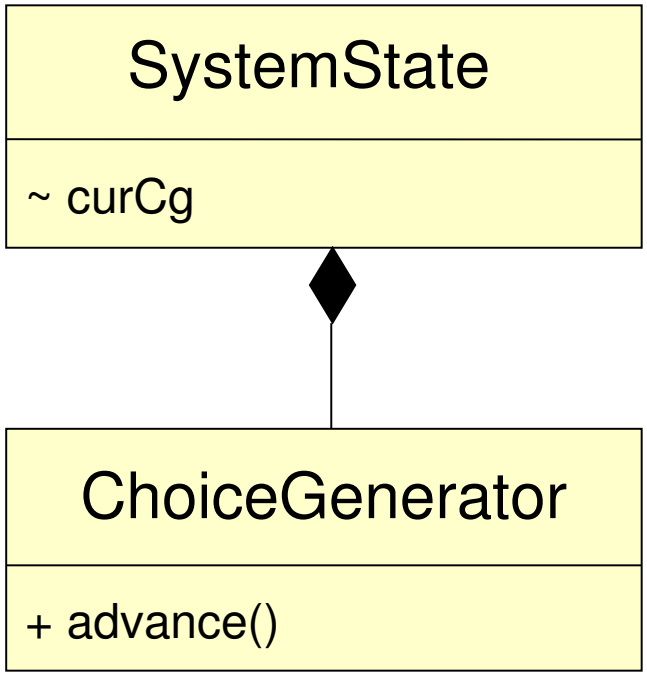


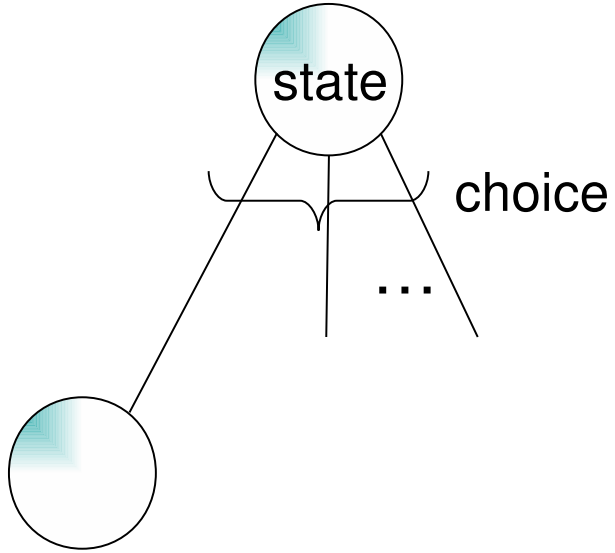
State Space

- By executing, state space is generated.
- State
  - JVM state
    - Info of each thread, Static info, Dynamic info
  - Choices
    - Scheduling choice
    - Data choice
- Transition
  - Sequence of instructions leads from one state to another

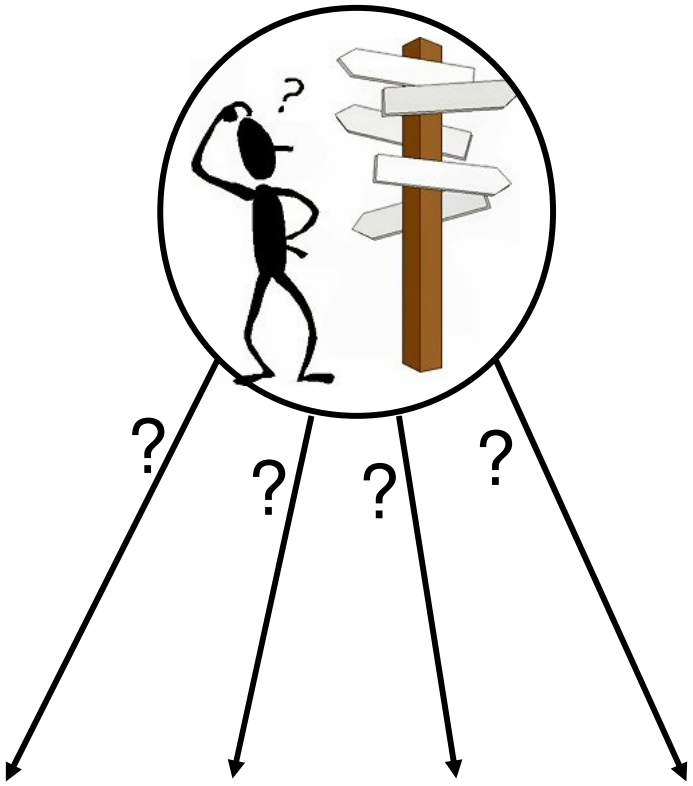


ChoiceGenerator and SystemState



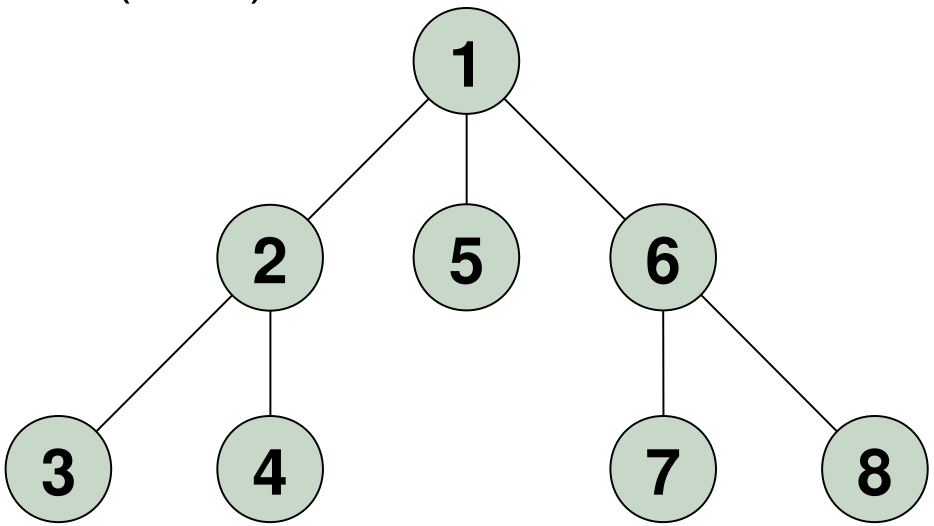


Which way to go?



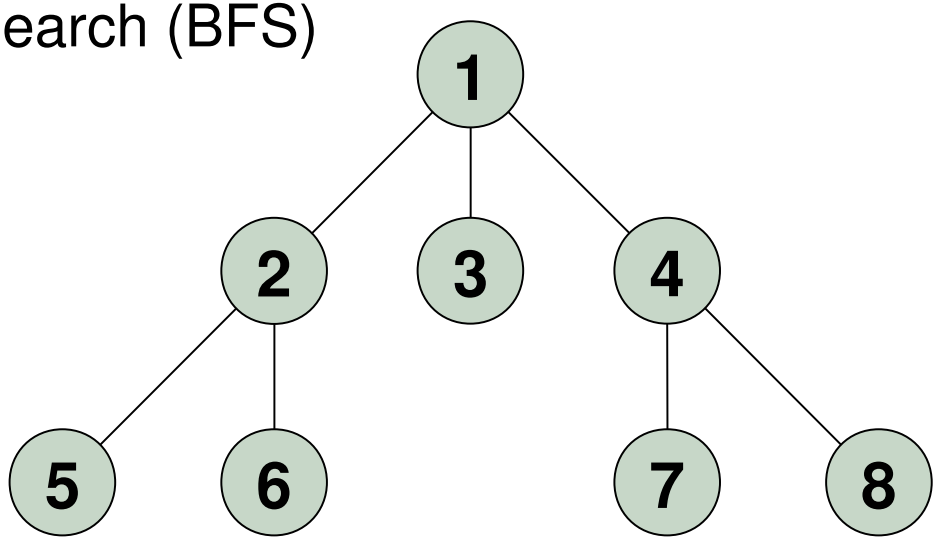
Search Component

- Search: driver for JVM
- Responsible for selecting the next state
- Different search algorithm
  - Depth-First Search (DFS)



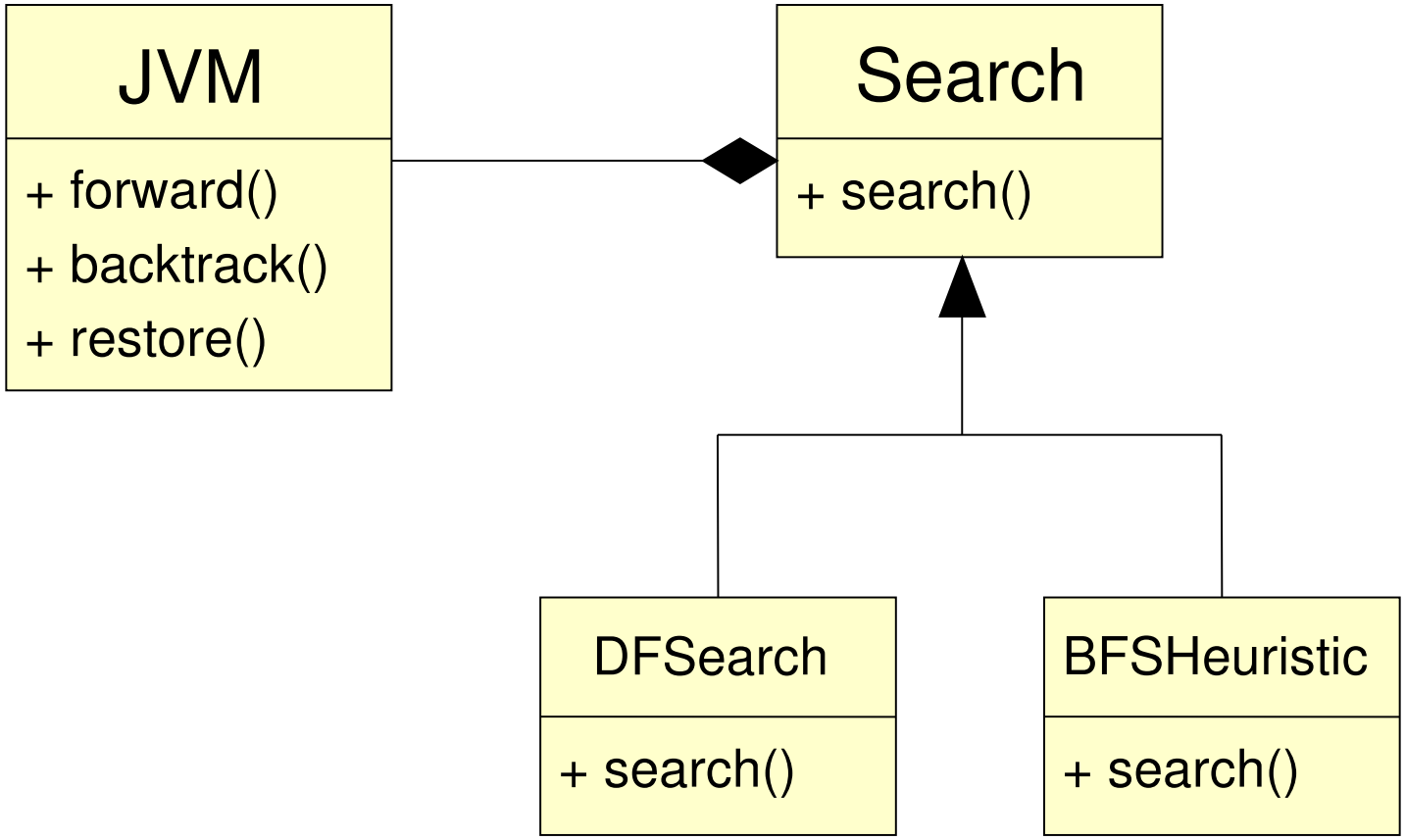
Search Component

- Breadth-First Search (BFS)



- RandomSearch: behaves like a normal JVM and searches until certain number of execution
- Heuristic: i.e. gives priority to the states
- …

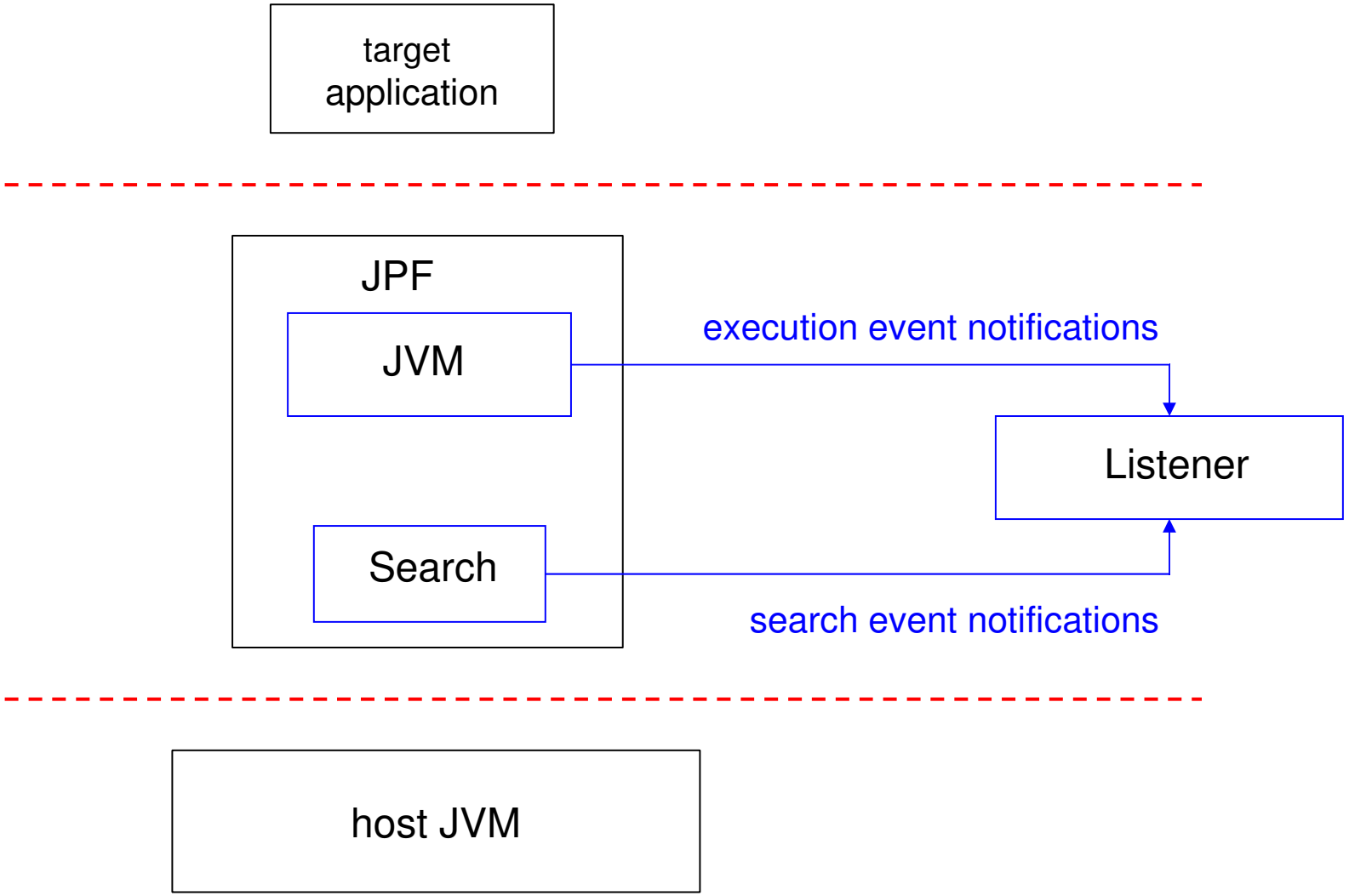
JVM and Search Classes



Listeners

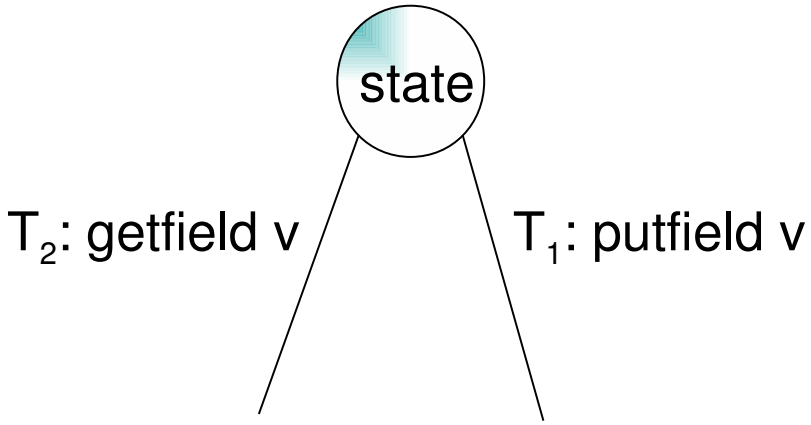
- Using listeners, JPF can be easily extended
- JPF notifies listeners about certain events
- Main interfaces: VMListener, SearchListener
- Wide range of events are considered
  - instructionExecuted,
  - stateAdvanced,
  - choiceGeneratorSet,
  - searchStarted,
  - searchFinished,
  - …
- Example: race detectors

Listeners

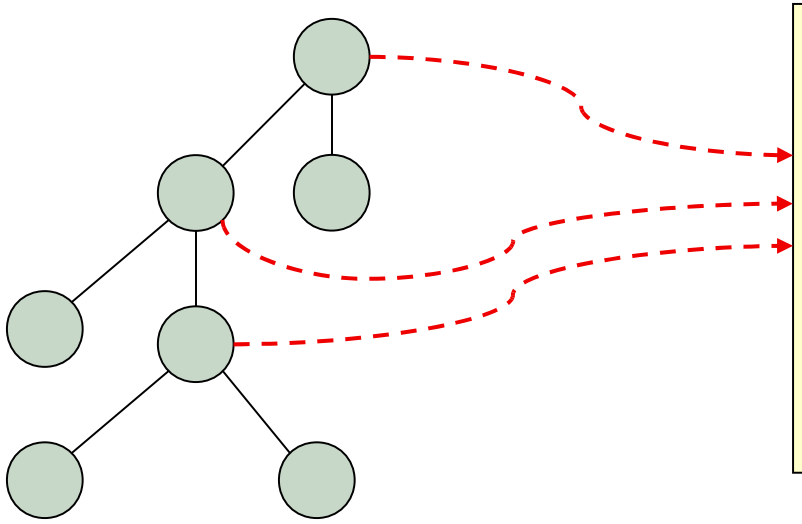

target application
JPF
JVM
Search
execution event notifications
Listener
search event notifications
host JVM

PreciseRaceDetector

- Used to detect (data) races
- What is race?
  - Arises in concurrent programs
  - Simultaneous access to a variable
  - Access is conflicting:
    - At least one write access



PreciseRaceDetector



```
public class PreciseRaceDetector extends
PropertyListenerAdapter
{
   . . .
   public void choiceGeneratorSet(JVM vm)
   {
      . . .
   }
   . . .
}
```

Race occurs if

- There is scheduling choice
- Threads, T_1 and T_2, are accessing a shared variable
- (T_1.isWriting() || T_2.isWriting())

jpf.properties

- Used to change the JPF configuration from default
- Property: key = value
  - e.g. set search to RandomSearch:

    `search.class=gov.nasa.jpf.search.RandomSearch`

  - e.g. make JPF to find races:

    `listener=gov.nasa.jpf.listener.PreciseRaceDetector`

More about JPF Project…

- jpf-core:
  - containing the basic VM and modelchecking infrastructure.
  - Number of classes: 945
  - Number of source lines (without comments): 47190
- How to run JPF using command line?
  - jpf +classpath=. targetClassName
  - Do NOT run JPF on indigo or red
- Where to put your jpf.properties file?
  - In the directory that you run jpf

Examples

User-defined assertion

- Example:

```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=1

User-defined assertion

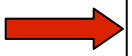
- Example:

```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=1
b=false

User-defined assertion

- Example:



```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=1
b=false

User-defined assertion

- Example:

```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=1
b=false

User-defined assertion

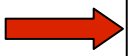
- Example:

```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=1
b=true

User-defined assertion

- Example:



```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=1
b=true

User-defined assertion

- Example:

```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=0
b=true

User-defined assertion

- Example:

```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

i=1
j=0
b=true

User-defined assertion

- Example:

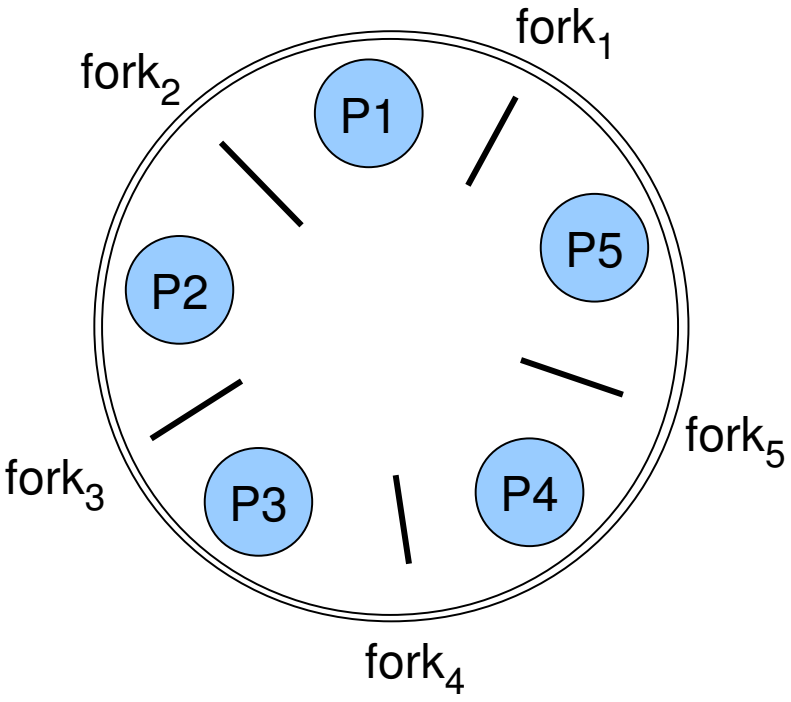
```
int i = 1, j = 1;
boolean b  =
    (new Random()).nextBoolean();
if(b)
    j = 0
assert(i==j);
```

ERROR!

i=1
j=0
b=true

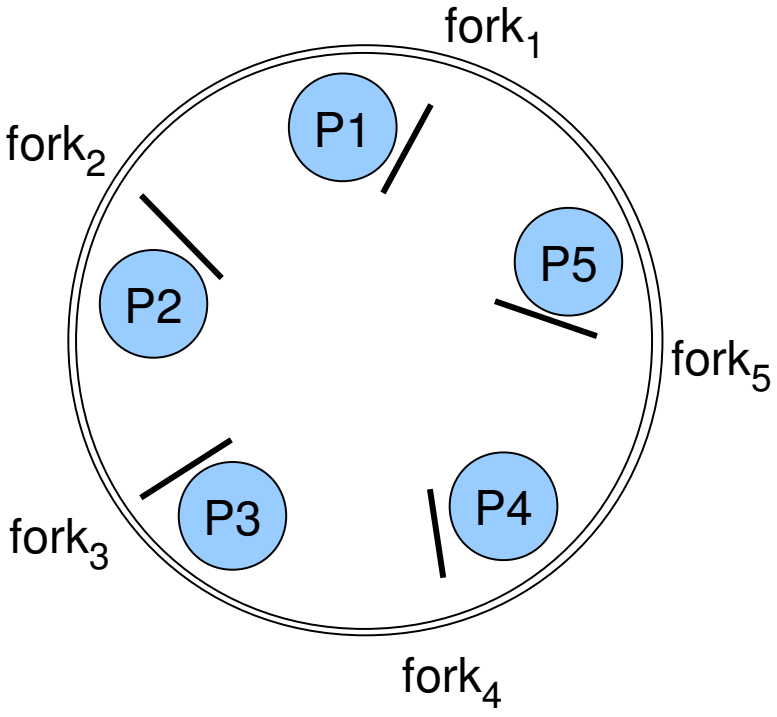
Deadlock

- Example: Dining Philosophers



Deadlock

- Example: Dining Philosophers



Deadlock

- Example: Dining Philosophers