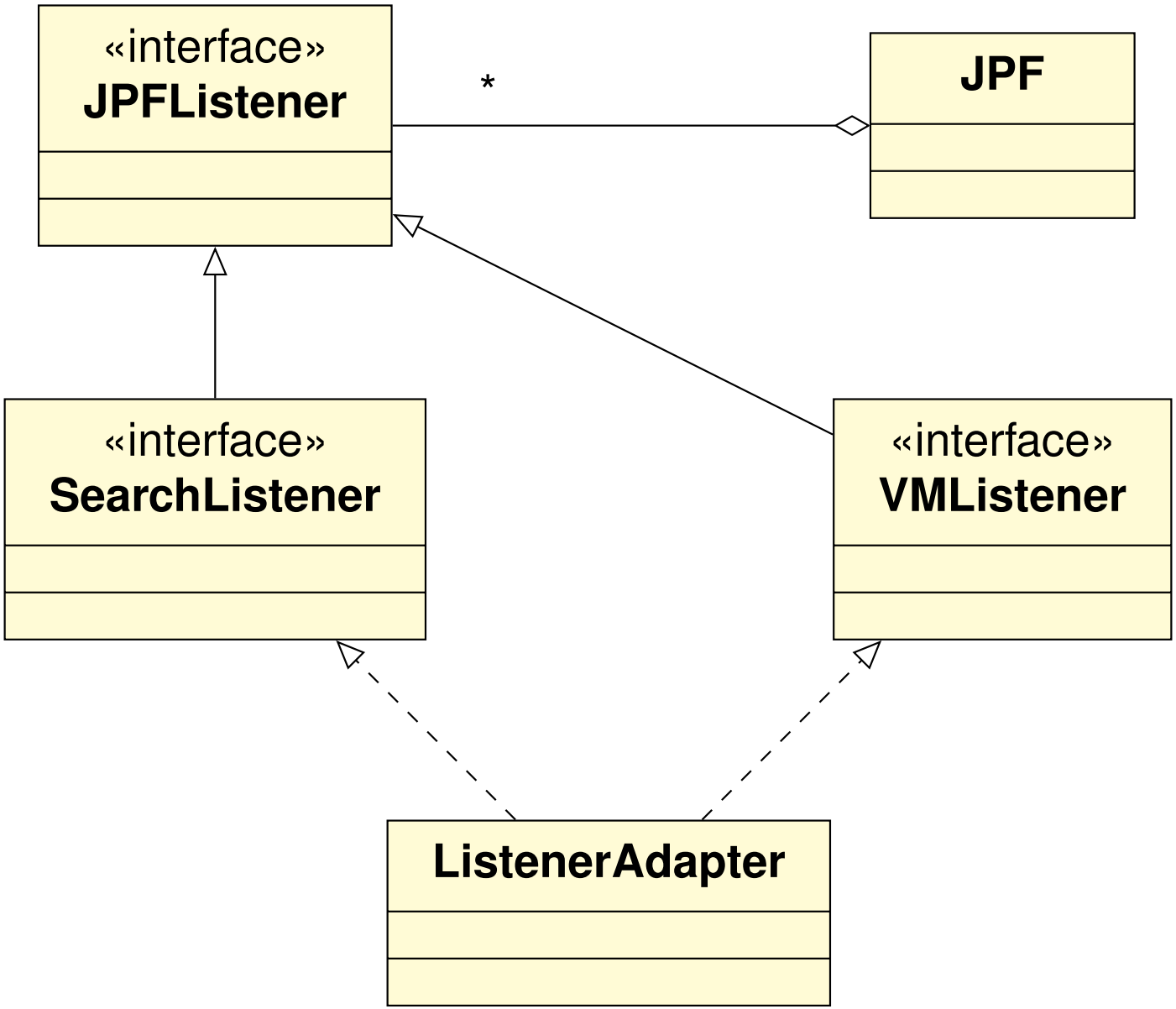
Listen!

EECS 4315

www.cse.yorku.ca/course/4315/

Generator and Listener



JPFListener

The interface `JPFListener` is empty.

JPFListener

The interface `JPFListener` is empty.

Question

Why introduce an empty interface?

JPFListener

The interface `JPFListener` is empty.

Question

Why introduce an empty interface?

Answer

For type checking.

VMListener

```
public interface VMListener extends JPFListener {

  // VM has been initialized and, hence, classes
  // have been loaded
  void vmInitialized(VM vm);

  // A number of methods related to the execution
  // of instructions
  void executeInstruction(VM vm,
    ThreadInfo currentThread,
    Instruction instructionToExecute);
  ...

  // A number of methods related to threads
  void threadStarted(VM vm,
    ThreadInfo startedThread);
  ...
```

VMListener

```
// Class has been loaded
void loadClass(VM vm, ClassFile cf);

// A number of methods related to objects
void objectCreated(VM vm, ThreadInfo currentThrea
  ElementInfo newObject);
...

// A number of methods related to garbage
// collection
void gcBegin(VM vm);
...
```

VMListener

```
// A number of methods related to exceptions
void exceptionThrown(VM vm,
  ThreadInfo currentThread,
  ElementInfo thrownException);
...

// A number of methods related to choice
// generators
void choiceGeneratorRegistered(VM vm,
  ChoiceGenerator<?> nextCG,
  ThreadInfo currentThread,
  Instruction executedInstruction);
...
```

VMListener

```
  // A number of methods about methods
  void methodEntered(VM vm,
    ThreadInfo currentThread,
    MethodInfo enteredMethod);
  ...
}
```

Garbage Collector

Write a listener that print a $*$ whenever the garbage collector is invoked by JPF.

Garbage Collector

Write a listener that measures the amount of time (in milliseconds) JPF's garbage collector takes whenever it is invoked by JPF.

SearchListener

```
public interface SearchListener extends JPFListener
  void stateAdvanced(Search search);
  void stateProcessed(Search search);
  void stateBacktracked(Search search);
  void statePurged(Search search);
  void stateStored(Search search);
  void stateRestored(Search search);
  void propertyViolated(Search search);
  void searchStarted(Search search);
  void searchConstraintHit(Search search);
  void searchFinished(Search search);
}
```

State Space

Implement a listener which prints the states and transitions visited by the search in the following simple format:

```
0 -> 1
1 -> 2
0 -> 3
3 -> 4
4 -> 2
```

State Space

Implement a listener which creates a dot file representing the the states and transitions visited by the search.

```
digraph statespace {
0 -> 1
1 -> 2
0 -> 3
3 -> 4
4 -> 2
}
```