Quiz 2

- You can find your grade for Quiz 2 at `https://www.cse.yorku.ca/~roumani/ePost/server/ep.cgi?year=2017-18&term=W&course=4315`.
- You received an email with feedback at your EECS account.

Draft project proposal

Please submit a draft of your project proposal (worth 2%) before Wednesday February 21 by

- transferring the file to `red.eecs.yorku.ca` and
- submitting the file using

  ```
  submit 4315 draft <name of file>
  ```

Search

EECS 4315

www.eecs.yorku.ca/course/4315/

Source: weknowyourdreams.com

Testing our searches

Question

How do we test our `DFSearch` and `BFSearch`?

Testing our searches

Question

How do we test our `DFSearch` and `BFSearch`?

Answer

Compare them with the corresponding JPF search strategies.

Testing our searches

Question

How do we test our `DFSearch` and `BFSearch`?

Answer

Compare them with the corresponding JPF search strategies.

Question

How do we compare search strategies?

Testing our searches

Question

How do we test our `DFSearch` and `BFSearch`?

Answer

Compare them with the corresponding JPF search strategies.

Question

How do we compare search strategies?

Answer

Implement a search listener that records the notifications.

Testing our searches

Question

How do we represent a notification of a search?

Testing our searches

Question

How do we represent a notification of a search?

Answer

For example, as a `String`.

Testing our searches

Question

How do we represent a notification of a search?

Answer

For example, as a `String`.

Question

How do we represents a collection of notifications?

Testing our searches

Question

How do we represent a notification of a search?

Answer

For example, as a `String`.

Question

How do we represents a collection of notifications?

Answer

For example, as a `List<String>` (order in which the notifications happen matters).

SearchNotificationRecorder

```
public class SearchNotificationRecorder
  implements SearchListener {

  private List<String> notifications;

  public SearchNotificationRecorder() {
    this.notification = new ArrayList<String>();
  }

  ...
}
```

SearchNotificationRecorder

Question

Implement

```
public void searchStarted(Search search) {
  ...
}
```

SearchNotificationRecorder

Question

Implement

```
public void searchStarted(Search search) {
  ...
}
```

Answer

```
public void searchStarted(Search search) {
  this.notifications.add("started in state "
                          + search.getStateId());
}
```

SearchNotificationRecorder

Question

Implement

```
public void stateAdvanced(Search search) {
  ...
}
```

SearchNotificationRecorder

Question

Implement

```
public void stateAdvanced(Search search) {
  ...
}
```

Answer

```
public void stateAdvanced(Search search) {
  this.recording.add("advanced to state "
                     + search.getStateId());
}
```

SearchNotificationRecorder

Question

In which method is the list serialized?

SearchNotificationRecorder

Question

In which method is the list serialized?

Answer

In `searchFinished`.[a]

[a] One might be tempted to use Java's `finalize` method. However, JPF does not ensure that search listeners can be garbage collected at the end of the search and, hence, the `finalize` method is not invoked.

SearchNotificationRecorder

Question

In which method is the list serialized?

Answer

In `searchFinished`.[a]

[a] One might be tempted to use Java's `finalize` method. However, JPF does not ensure that search listeners can be garbage collected at the end of the search and, hence, the `finalize` method is not invoked.

Question

What should we do if a notification occurs after `searchFinished`?

SearchNotificationRecorder

Question

In which method is the list serialized?

Answer

In `searchFinished`.[a]

[a] One might be tempted to use Java's `finalize` method. However, JPF does not ensure that search listeners can be garbage collected at the end of the search and, hence, the `finalize` method is not invoked.

Question

What should we do if a notification occurs after `searchFinished`?

Answer

For example, serialize the list again.

Serialize the list

```
private void serialize() {
  try {
    FileOutputStream output
      = new FileOutputStream("notifications.ser");
    ObjectOutputStream stream
      = new ObjectOutputStream(output);
    stream.writeObject(this.notifications);
    stream.close();
    output.close();
  } catch (IOException e) {
    System.out.println("Something went wrong with serializi
  }
}
```

Serialize the list

We want the user to be able to specify the name of the file to store the serialized list.

1. Add a key and corresponding value for the file name in the configuration file.
2. Extract the file name from the `Config` object in the constructor.
3. Store the file name in an attribute.
4. Use the attribute in the `serialize` method.

Serialize the list

Add a key and corresponding value for the file name in the configuration file.

```
...
listener=SearchNotificationRecorder
recorder.file=notifications.ser
...
```

Serialize the list

Extract the file name from the `Config` object in the constructor.

```
public SearchNotificationRecorder(Config config) {
  ...
  String fileName
    = config.getString("recorder.file", "tmp.ser");
  ...
}
```

Serialize the list

Store the file name in an attribute.

```
public class SearchNotificationRecorder
  implements SearchListener {

  private String fileName;

  public SearchNotificationRecorder(Config config) {
    ...
    this.fileName
      = config.getString("recorder.file", "tmp.ser");
    ...
  }

  ...
}
```

Serialize the list

Use the attribute in the serialize method.

```
private void serialize() {
  try {
    FileOutputStream output
      = new FileOutputStream(this.fileName);
    ObjectOutputStream stream
      = new ObjectOutputStream(output);
    stream.writeObject(this.recording);
    stream.close();
    output.close();
  } catch (IOException e) {
    System.out.println("Something went wrong with serializi
  }
}
```

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

How do we keep track whether `searchFinished` has been invoked?

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

How do we keep track whether `searchFinished` has been invoked?

Answer

Introduce an attribute `finished`.

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

Add attribute `finished`.

Serialize the list

We call serialize in searchFinished. We serialize the list again if a notification occurs after searchFinished.

Question

Add attribute finished.

Answer

```
private boolean finished;
```

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

Initialize attribute `finished`.

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

Initialize attribute `finished`.

Answer

```
public SearchNotificationRecorder() {
  ...
  this.finished = false;
  ...
}
```

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

Implement `searchFinished`.

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

Implement `searchFinished`.

Answer

```
public void searchFinished(Search search) {
  this.recording.add("finished");
  this.finished = true;
  this.serialize();
}
```

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

Implement `searchStarted`.

Serialize the list

We call `serialize` in `searchFinished`. We serialize the list again if a notification occurs after `searchFinished`.

Question

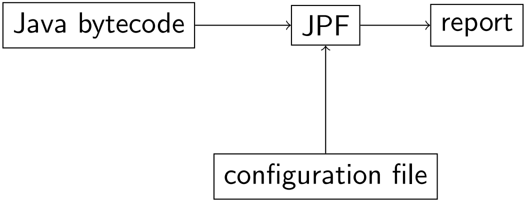
Implement `searchStarted`.

Answer

```
public void searchStarted(Search search) {
  this.recording.add("started");
  this.checkFinished();
}

private checkFinished() {
  if (this.finished) {
    this.serialize();
  }
}
```

JPF



configuration:

```
listener=SearchNotificationRecorder
recorder.file=notifications.ser
```

report:

`notifications.ser` contains a serialized list of notifications

Code

Question

For which (byte)code should we run JPF with the `SearchNotificationRecorder` listener?

Code

Question

For which (byte)code should we run JPF with the `SearchNotificationRecorder` listener?

Answer

"Random" (byte)code.

Code

Question

Given a finite directed graph G, generate a Java app such that JPF run on the app with the `SimpleDot` listener produces G.

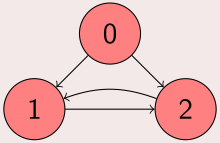
Code

Question

Given a finite directed graph G, generate a Java app such that JPF run on the app with the `SimpleDot` listener produces G.

Question

Given a finite directed graph



$$\begin{tikzcd} & 0 \arrow[dl] \arrow[dr] & \\ 1 \arrow[rr] & & 2 \arrow[ll, bend right] \end{tikzcd}$$

generate a Java app such that JPF run on the app with the `SimpleDot` listener produces G.

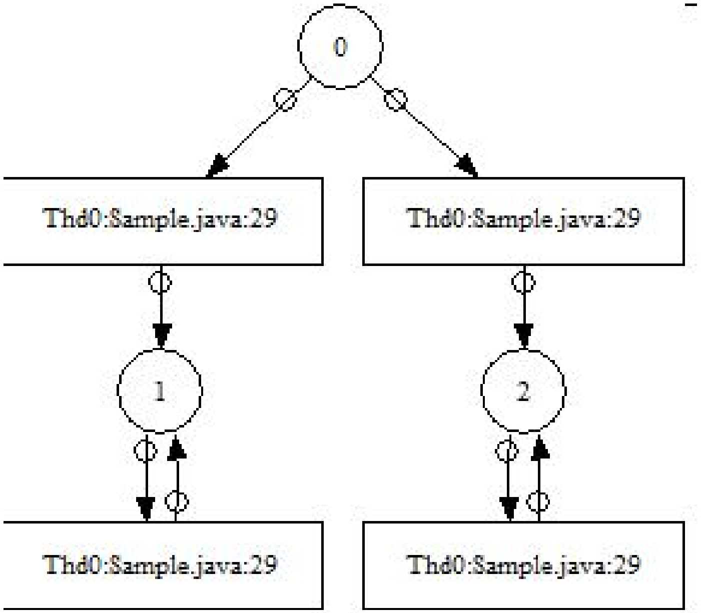
Code

```java
public class Sample {
  public static void main(String[] args) {
    final Random RANDOM = new Random();
    boolean done = false;
    int state = 0;
    while (!done) {
      switch (state) {
        ...
      }
    }
  }
}
```

Code

```
case 0:
  switch (RANDOM.nextInt(2)) {
    case 0:
      state = 1; break;
    case 1:
      state = 2; break;
  };
  break;
case 1:
  switch (RANDOM.nextInt(1)) {
    case 0:
      state = 2; break;
  };
  break;
case 2:
  switch (RANDOM.nextInt(1)) {
    case 0:
      state = 1; break;
  };
  break;
```

State space


0
Thd0:Sample.java:29
Thd0:Sample.java:29
1
2
Thd0:Sample.java:29
Thd0:Sample.java:29

Code

`name`: name of the app

`number`: number of states of the model of the app

Question

Generate a class with name `name` such that its model is a random graph with `number` vertices.

Code

Answer

```
1 open file name for writing
2 print line 1–7
3 for $n = 0, \ldots,$ number
    1 print case n:
    2 choose successors of state $n$ randomly
    3 determine the number of successors
    4 if successors $= 0$ then
        1 print done = true;
      else
        1 print switch (RANDOM.nextInt(successors)) {
        2 for $s = 0, \ldots,$ successors
            1 print case s:
            2 print state = i; break, where $i$ is the $s$th successor
        3 print }
4 print line 28–31
```

Shell script

```
# Generate code
java Generate Sample.java 5

# Compile code
javac Sample.java

# Run JPF with gov.nasa.jpf.search.heuristic.BFSHeuristic
java -cp /cs/fac/packages/jpf/jpf-core/build/jpf.jar gov.nasa.jpf.JPF \
+target=Sample \
+classpath=. \
+native_classpath=. \
+cg.enumerate_random=true \
+search.class=gov.nasa.jpf.search.heuristic.BFSHeuristic \
+listener=SearchNotificationRecorder \
+recorder.file=first.ser

# Run JPF with BFSearch
java -cp /cs/fac/packages/jpf/jpf-core/build/jpf.jar gov.nasa.jpf.JPF \
+target=Sample \
+classpath=. \
+native_classpath=. \
+cg.enumerate_random=true \
+search.class=BFSearch \
+listener=SearchNotificationRecorder \
+recorder.file=second.ser

# Compare recordings
java CompareSearchRecordings first.ser second.ser
```