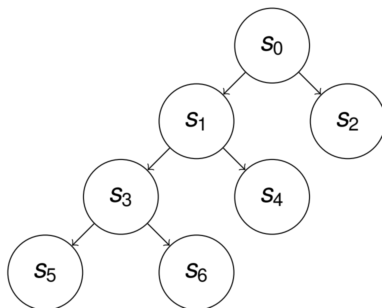
Listen!

EECS 4315

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

Search



Task 1

Develop a Java app that prints output. When checking the Java app by JPF with depth-first search (DFS) the output should be different from the output for breadth-first search (BFS).

Search

```
Random random = new Random();
System.out.println("s0");
if (random.nextBoolean()) {
  System.out.println("s2");
} else {
  System.out.println("s1");
  if (random.nextBoolean()) {
    System.out.println("s4");
  } else {
    System.out.println("s3");
    if (random.nextBoolean()) {
      System.out.println("s6");
    } else {
      System.out.println("s5");
    }
  }
}
```

Task 2

Verify your program using JPF with BFS and DFS. To do that, you need to create an application properties file (.jpf file) for your Java app developed in Task 1. Configure the search property to be gov.nasa.jpf.search.heuristic.BFSHeuristic or gov.nasa.jpf.search.heuristic.DFSHeuristic.

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search=gov.nasa.jpf.search.heuristic.BFSHeuristic
```

```
=====================================================
Traversal.main()

=====================================================
s0
s1
s3
s5
s6
s4
s2

=====================================================
```

Search

That is not breadth first search!

Question

Have we set the search property correctly? How can we check that?

That is not breadth first search!

Question

Have we set the search property correctly? How can we check that?

Answer

Use the following command line arguments

- `-log`: lists the order in which properties files got loaded
- `-show`: prints all configuration entries after the initialization is complete

Log Option

```
loading property file: ...\.jpf\site.properties
loading property file: ...\jpf\jpf-core\jpf.propert
collected native_classpath=...\jpf\jpf-core/build/j
collected native_libraries=null
```

Show Option

```
...
search = gov.nasa.jpf.search.heuristic.BFSHeuristic
search.class = gov.nasa.jpf.search.DFSearch
...
```

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.heuristic.BFSHeuri
```

```
==================================================================
Traversal.main()

==================================================================
s0
s1
s2
s3
s4
s5
s6

==================================================================
```

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.heuristic.DFSHeuri
```

```
==================================================
Traversal.main()

==================================================
s0
s1
s2
s3
s4
s5
s6

==================================================
```

That is not depth first search!

Search

That is not depth first search!

Let's try instead gov.nasa.jpf.search.DFSearch.

Search

```
target=Traversal
classpath=.
cg.enumerate_random=true
search.class=gov.nasa.jpf.search.DFSearch
```

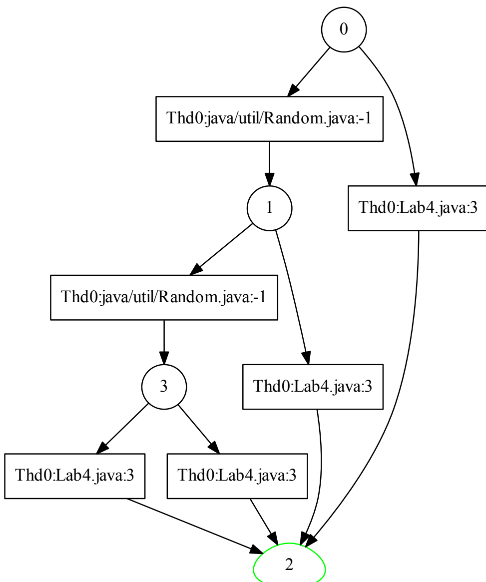
Search

```
======================================================
Traversal.main()

======================================================
s0
s1
s3
s5
s6
s4
s2

======================================================
```

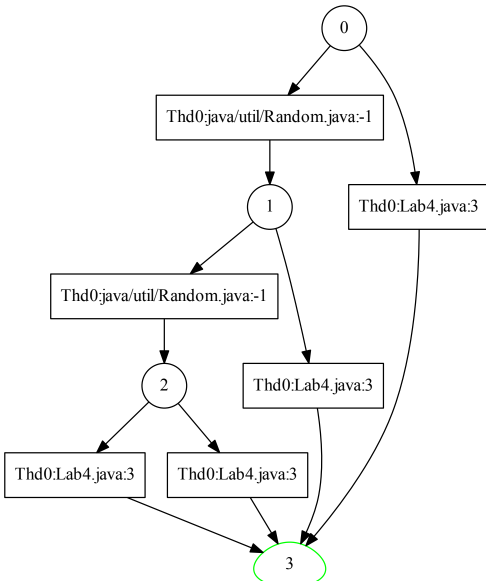
Task 3

Generate the state space diagram for BFS and DFS. To do this you need to set the listener to StateSpaceDot.

BFS



DFS



Task 4

Verify your program using RS. RS can explore several random executions and in JPF you have the freedom to set the maximum number of executions you would like RS to explore. Firstly, set your search strategy to gov.nasa.jpf.search.RandomSearch. Secondly, set the search.RandomSearch.path_limit property to be any integer larger than 0. Compare the resulting state space diagrams.

```
JavaPathfinder core system v8.0 (rev 2+) - (C) 2005

======================================================
Traversal.main()

======================================================
s0
s1
s3
s5
```

Search

No space diagram has been produced.

Search

No space diagram has been produced.

Bugs are everywhere, even in JPF!

Event Listeners

JPF uses (event) listeners.

```
target=Traversal
classpath=.
cg.enumerate_random=true
listener=gov.nasa.jpf.listener.StateSpaceDot
```

Event Generator

The method `run` of the class `Generator` produces integer values. On average, it produces an integer value every two seconds (according to a Gaussian distribution with a mean of two seconds and a standard deviation of one second). It produces integers in the interval [0, 9] uniformly at random.

Event Generator

```
public class Generator {
  public void run() {
    Random random = new Random();
    final int MEAN_DELAY = 2000;
    final int SD_DELAY = 1000;
    final int MAX_VALUE = 9;

    while (true) {
      int delay = MEAN_DELAY +
        (int) (SD_DELAY * random.nextGaussian());
      Thread.sleep(delay);
      int value = random.nextInt(MAX_VALUE + 1);
    }
  }
}
```

Main App

The `Main` app creates a `Generator` object and invokes its `run` method.

Main App

```
public class Main {
  public static void main(String[] args) {
    Generator generator = new Generator();
    generator.run();
  }
}
```

Event Listener

Whenever the `Generator` produces an integer, we want to process it. For example, we can print $*$. We want to decouple the processing of the integers from the production of the integers so that we need not make any changes to the `Generator` class if we want to change the processing of the integers. Hence, we create a `StarPrinter` class with a method `process` to print $*$.

Event Listener

```
public class StarPrinter {
  public void process() {
    System.out.println("*");
  }
}
```

Event Generator and Listener

Whenever the `Generator` produces an integer, it should invoke the `process` method on a `StarPrinter` object.

```
public class Generator {
  public void run() {
    ...
    while (true) {
      int delay = ...
      Thread.sleep(delay);
      int value = random.nextInt(...);
      ???.process();
    }
  }
}
```

Event Generator and Listener

Question

How do we store the reference `???` to a `StarPrinter` object in the `Generator` class?

Event Generator and Listener

Question

How do we store the reference `???` to a `StarPrinter` object in the `Generator` class?

Answer

As an attribute.

Event Generator and Listener

Question

How do we store the reference `???` to a `StarPrinter` object in the `Generator` class?

Answer

As an attribute.

```
public class Generator {
  private ??? x;

  public void run() {
    ...
    while (true) {
      ...
      this.x.process();
    }
  }
```

Event Generator and Listener

Question

What is the type of the attribute `x`?

Event Generator and Listener

Question

What is the type of the attribute `x`?

Answer

`StarPrinter`.

Event Generator and Listener

Question

What is the type of the attribute `x`?

Answer

`StarPrinter`.

```
public class Generator {
  private StarPrinter x;

  public void run() {
    ...
    while (true) {
      ...
      this.x.process();
    }
  }
}
```

Event Generator and Listener

```
public class PlusPrinter {
  public void process() {
    System.out.println("+");
  }
}
```

Event Generator and Listener

```
public class PlusPrinter {
  public void process() {
    System.out.println("+");
  }
}
```

Question

How can we modify the type of the attribute `x` and the classes `StarPrinter` and `PlusPrinter` so that the class `Generator` can use both?

Event Generator and Listener

```
public class PlusPrinter {
  public void process() {
    System.out.println("+");
  }
}
```

Question

How can we modify the type of the attribute `x` and the classes `StarPrinter` and `PlusPrinter` so that the class `Generator` can use both?

Answer

Introduce an interface `Listener`, change the type of the attribute `x` to `Listener`, and specify that the classes `StarPrinter` and `PlusPrinter` implement `Listener`.

Listener

```
public interface Listener {
  void process();
}
```

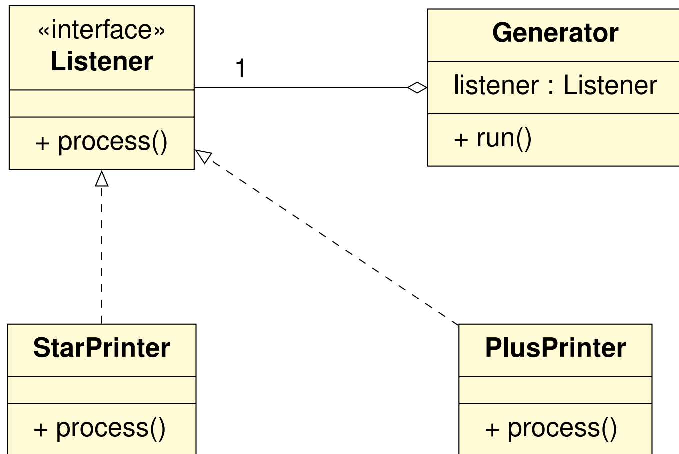
Generator

```
public class Generator {
  private Listener listener;

  public void run() {
    ...
    while (true) {
      ...
      this.listener.process();
    }
  }
}
```

Listener

```
public class StarPrinter implements Listener {
  public void process() {
    System.out.println("*");
  }
}
```

Generator and Listener



Question

How do we initialize the `listener` attribute of the `Generator` class?

Question

How do we initialize the `listener` attribute of the `Generator` class?

Question

By means of a mutator `setListener`.

Generator

Question

How do we initialize the `listener` attribute of the `Generator` class?

Question

By means of a mutator `setListener`.

```
public class Generator {
  private Listener listener;

  public void setListener(Listener) {
    this.listener = listener;
  }
}
```

Main App

```
public class Main {
  public static void main(String[] args) {
    Generator generator = new Generator();
    Listener starPrinter = new StarPrinter();
    generator.setListener(startPrinter);
    generator.run();
  }
}
```

Multiple Listeners

Question

Which changes do we have to make if we want to associate multiple listeners with the generator? For example, we would like a $*$ and + to be printed whenever an integer is produced.

Multiple Listeners

Question

Which changes do we have to make if we want to associate multiple listeners with the generator? For example, we would like a $*$ and $+$ to be printed whenever an integer is produced.

Answer

Instead of an attribute that represents a `Listener`, use an attribute that represents a collection of `Listener`s.

Multiple Listeners

Question

Instead of

```
private Listener listener;
```

what do we use to represent a collection of `Listener`s?

Multiple Listeners

Question

Instead of

```
private Listener listener;
```

what do we use to represent a collection of `Listener`s?

Question

```
private List<Listener> listeners;
```

Question

Where and how do we initialize the attribute `listeners`?

Multiple Listeners

Question

Where and how do we initialize the attribute `listeners`?

Question

```
public Generator() {
  this.listeners = new ArrayList<Listener>;
}
```

Multiple Listeners

Question

How do we add a listener to the `listeners`?

Multiple Listeners

Question

How do we add a listener to the `listeners`?

Question

```
public void addListener(Listener listener) {
  this.listeners.add(listener);
}
```

Multiple Listeners

Question

How do we invoke the `process` method on the `listeners`?

Multiple Listeners

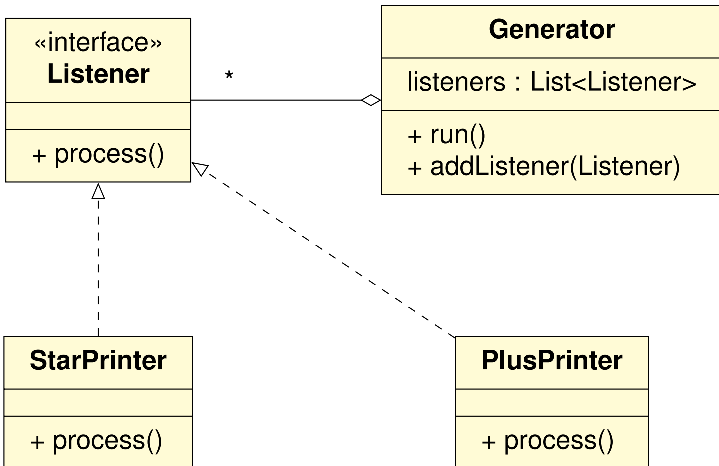
Question

How do we invoke the `process` method on the `listeners`?

Question

```
int value = random.nextInt(...);
for (Listener listener : this.listeners) {
  listener.process();
}
```

Generator and Listener



Listener

Whenever the `Generator` produces an integer, we want to print it.

Listener

Whenever the `Generator` produces an integer, we want to print it.

Question

How does the `Generator` pass the produced integer to the `Listener`?

Listener

Whenever the `Generator` produces an integer, we want to print it.

Question

How does the `Generator` pass the produced integer to the `Listener`?

Answer

Pass the produced integer as an argument.

```
public void process(int value) {
  ...
}
```

Listener

Whenever the `Generator` terminates, we want to print the sum of the integers it produced.

Listener

The `run` method of the `Generator` class is modified as follows.

```
final int STOP = 5;
boolean done = false;
while (!done) {
  ...
  done = random.nextInt(STOP) == 0;
}
```

Listener

Whenever the `Generator` terminates, we want to print the sum of the integers it produced.

Question

Which changes have to be made to the `Listener` interface?

Listener

Whenever the `Generator` terminates, we want to print the sum of the integers it produced.

Question

Which changes have to be made to the `Listener` interface?

Answer

Add

```
void stop();
```

Listener

Whenever the `Generator` terminates, we want to print the sum of the integers it produced.

Question

Which changes have to be made to the `Generator` class?

Listener

Whenever the `Generator` terminates, we want to print the sum of the integers it produced.

Question

Which changes have to be made to the `Generator` class?

Answer

```
final int STOP = 5;
boolean done = false;
while (!done) {
  ...
  done = random.nextInt(STOP) == 0;
}
for (Listener listener : this.listeners) {
  listener.stop();
}
```

Listener

Whenever the `Generator` terminates, we want to print the sum of the integers it produced.

Question

Which changes have to be made to the `ListenerAdapter` class?

Listener

Whenever the `Generator` terminates, we want to print the sum of the integers it produced.

Question

Which changes have to be made to the `ListenerAdapter` class?

Answer

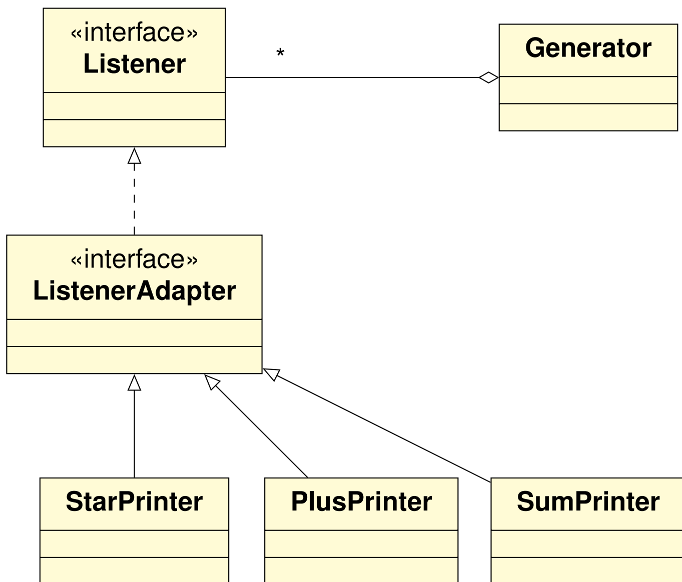
Add

```
public void stop() {
  // default implementation
}
```

SumPrinter

```
public class SumPrinter extends ListenerAdapter {
  private int sum;

  public SumPrinter() {
    this.sum = 0;
  }

  public void process(int value) {
    this.sum += value;
  }

  public void stop() {
    System.out.println("---------------------");
    System.out.println(this.sum);
    System.out.println("---------------------");
  }
}
```

Generator and Listener



Quiz

There will be a quiz on Friday February 3 in class about the material covered during the period January 13–January 25.