Concurrency

EECS 4315

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

Race condition and data race

A race condition is a flaw that occurs when the timing or ordering of events affects a program's correctness. Generally speaking, some kind of external timing or ordering non-determinism is needed to produce a race condition.

A data race happens when there are two memory accesses in a program where both

- target the same location,
- are performed concurrently by two threads,
- are not all reads (at least one is a write),
- are not synchronization operations.

Race condition and data race

Many race conditions are due to data races, and many data races lead to race conditions. However, we can have race conditions without data races and data races without race conditions.

Race condition and data race

Many race conditions are due to data races, and many data races lead to race conditions. However, we can have race conditions without data races and data races without race conditions.

Question

Give an example that has both a data race and a race condition.

Hint

We have already seen such an example earlier in the course.

Race condition and data race

```
/**
 * Two threads, that share an account and both do
 * a deposit concurrently, cause a data race and
 * a race condition.
 */
public class Account {
  private double balance;

  public Account() {
    this.balance = 0;
  }

  public void deposit(double amount) {
    this.balance += amount;
  }
}
```

Race condition and data race

```
public class Customer extends Thread {
  private Account account;

  public Customer(Account account) {
    this.account = account;
  }

  public void run() {
    this.account.deposit(1);
  }
}
```

Race condition and data race

```
public class Customers {
  public static void main(String[] args) {
    Account account = new Account();
    (new Customer(account)).start();
    (new Customer(account)).start();
  }
}
```

Race condition and data race

JPF application properties file

```
target=Customers
classpath=<path to Customers.class>
listener=gov.nasa.jpf.listener.PreciseRaceDetector
```

Race condition and data race

Using JPF to detect data races.

Race condition and data race

JPF's output

```
JavaPathfinder core system v8.0 (rev 32+) - (C) 2005-2014 U

========================================================== syst
concurrency.Customers.main()

========================================================== sear

========================================================== erro
gov.nasa.jpf.listener.PreciseRaceDetector
race for field concurrency.Account@15e.balance
  Thread-1 at concurrency.Account.deposit(Account.java:7)
"  WRITE: putfield concurrency.Account.balance
  Thread-2 at concurrency.Account.deposit(Account.java:7)
"  READ: getfield concurrency.Account.balance
```

Race condition and data race

JPF application properties file

```
target=Customers
classpath=<path to Customers.class>
listener=gov.nasa.jpf.listener.PreciseRaceDetector
@using=jpf-visual
report.errorTracePrinter.property_violation=trace
report.publisher+=,errorTracePrinter
report.errorTracePrinter.class=ErrorTracePrinter
shell=.shell.basicshell.BasicShell
shell.panels+=,errorTrace
shell.panels.errorTrace=ErrorTracePanel
```

Race condition and data race

Trans.	main 0	Thread-1 1	Thread-2 2

0-2	`Account account = new Account();`
	`...`
	`(new Customer(account)).start();`		

3-4		`package concurrency;`
		`...`
		`this.balance += amount;`	

5	`}` `public class Customers {`		
6		`this.balance += amount;`	

7			`package concurrency;`
			`...`
			`this.balance += amount;`

8		`this.balance += amount;`	

Race condition and data race

Many race conditions are due to data races, and many data races lead to race conditions. However, we can have race conditions without data races and data races without race conditions.

Question

Give an example that has a race condition but does not have a data race.

Hint

Modify the previous example.

Race condition and data race

```
/**
 * Two threads, that share an account and both do
 * a deposit concurrently, cause a race condition
 * but no data race.
 */
public class Account {
  private double balance;

  public void deposit(double amount) {
    double temp;
    synchronized (this) {
      temp = this.balance;
    }
    temp += amount;
    synchronized (this) {
      this.balance = temp;
    }
```

Race condition and data race

Many race conditions are due to data races, and many data races lead to race conditions. However, we can have race conditions without data races and data races without race conditions.

Question

Give an example that has a data race but does not have a race condition.

Hint

Search for an element in an array.

Race condition and data race

```
/**
 * Multiple threads searching for an element in an
 * array may cause a data race but not a race condition.
 */
public class Search {
  private int[] collection;
  private boolean found;

  public Search(int[] collection) {
    this.collection = collection;
    this.found = false;
  }

  public void find(int from, int to, int element) {
    for (int i = from; i < to && !this.found; i++) {
    if (this.collection[i] == element) {
      this.found = true;
    }
  }
```

Race condition and data race

Extending JPF to detect data races.

SimpleRaceDetector

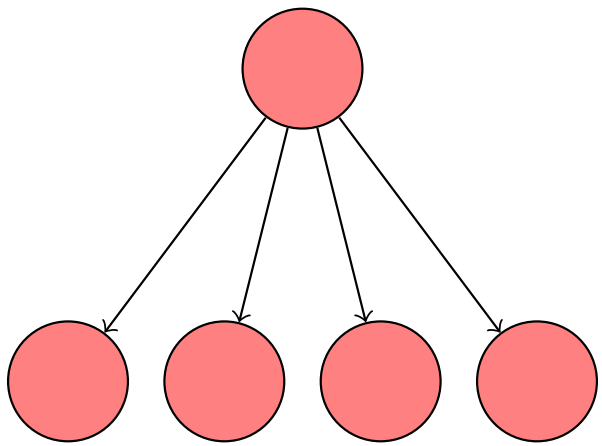
Problem

Develop a listener that detects data races (only for non-static attributes that are not arrays).

SimpleRaceDetector

Problem

Develop a listener that detects data races (only for non-static attributes that are not arrays).



There is a data race if in the first instruction of one transition thread t_1 accesses field f of object o and in the first instruction of another transition thread t_2 accesses field f of object o and $t_1 \neq t_2$ and at least one of the accesses is a write.

SimpleRaceDetector

Question

Fill in the dots.

```
for each branching state
  reads ← empty set
  writes ← empty set
  if the branching is caused by concurrency
    for each outgoing transition
      if the first instruction is a read
          of field f of object o
        if ...
          ...
        ...
      if the first instruction is a write
          of field f of object o
        if ...
          ...
        ...
```

SimpleRaceDetector

Answer

```
for each branching state
  reads ← empty set
  writes ← empty set
  if the branching is caused by concurrency
    for each outgoing transition
      if the first instruction is a read
          of field f of object o
        if writes contains [f, o]
          report data race
        add [f, o] to reads
      if the first instruction is a write
          of field f of object o
        if reads or writes contains [f, o]
          report data race
        add [f, o] to writes
```

SimpleRaceDetector

Question

Is it a `SearchListener` or a `VMListener`?

SimpleRaceDetector

Question

Is it a `SearchListener` or a `VMListener`?

Answer

A `VMListener`.

SimpleRaceDetector

Question

Is it a `SearchListener` or a `VMListener`?

Answer

A `VMListener`.

Question

By which class is branching caused by concurrency represented?

SimpleRaceDetector

Question

Is it a `SearchListener` or a `VMListener`?

Answer

A `VMListener`.

Question

By which class is branching caused by concurrency represented?

Answer

The `ChoiceGenerator` class.

SimpleRaceDetector

Question

Which methods of the `VMListener` interface should we implement?

SimpleRaceDetector

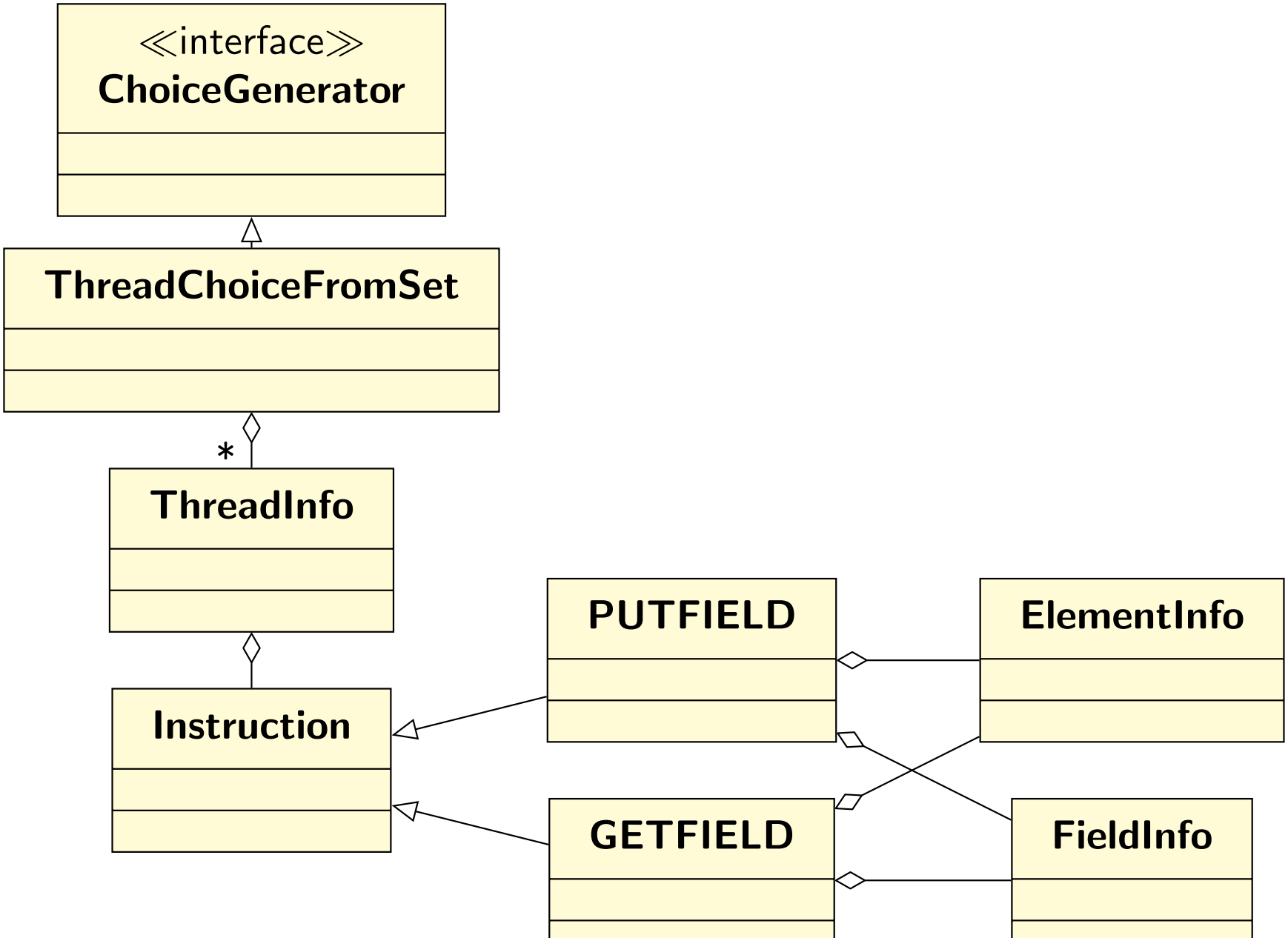
Question

Which methods of the `VMListener` interface should we implement?

Answer

The `choiceGeneratorSet` method.

JPF classes


≪interface≫
ChoiceGenerator
ThreadChoiceFromSet
*
ThreadInfo
Instruction
PUTFIELD
ElementInfo
GETFIELD
FieldInfo

SimpleRaceDetector

```
reads ← empty set
writes ← empty set
if the branching is caused by concurrency
  for each outgoing transition
```

```
public void choiceGeneratorSet(VM vm, ChoiceGenerator<?> choice)
  Set<Read> reads = new HashSet<Read>();
  Set<Write> writes = new HashSet<Write>();
  if (choice instanceof ThreadChoiceFromSet) {
    ThreadChoiceFromSet threadChoice =
      (ThreadChoiceFromSet) choice;
    for (ThreadInfo thread : threadChoice.getAllThreadChoices())
```

SimpleRaceDetector

```
if the first instruction is a read
  of field f of object o
if writes contains [f, o]
  report data race
add [f, o] to reads
```

```
Instruction instruction = thread.getPC();
if (instruction instanceof GETFIELD) {
  GETFIELD get = (GETFIELD) instruction;
  FieldInfo field = get.getFieldInfo();
  ElementInfo object = get.peekElementInfo(thread);
  Read read = new Read(object, field);
  if (writes.contains(read)) {
    System.out.printf("Data race on s\n", field);
  }
  reads.add(read);
}
```

Stack

Question

What are the two operations of the abstract data type Stack?

Stack

Question

What are the two operations of the abstract data type Stack?

Answer

push and pop.

Stack

We implement the stack as a singly linked list of nodes. Each node contains an element and a reference to the next node. The variable `top` refers to the first node of the linked list and is initially undefined (null).

Question

How can we implement the push operation?

Stack

We implement the stack as a singly linked list of nodes. Each node contains an element and a reference to the next node. The variable `top` refers to the first node of the linked list and is initially undefined (null).

Question

How can we implement the push operation?

Answer

```
new = node with element e;
new.next = top;
top = new;
```

Question

How can we implement the `pop` operation?

Question

How can we implement the pop operation?

Answer

```
if (top == null)
  return EMPTY;
else
  temp = top;
  top = top.next;
  return element of temp;
```

Compare-and-swap (CAS)

The operation CAS(variable, expected, new) atomically

- loads the value of variable,
- compares that value to expected,
- assigns new to variable if the comparison succeeds, and
- returns the old value of variable.

Compare-and-swap (CAS)

Assume that the shared variable `value` is initialized to zero.

```
Thread 1:
value = 1;
print "(1, " + value + ")";

Thread 2:
old = CAS(value, 0, 2);
print "(2, " + old + ")";
```

Question

What is the output produced and the final value of `value`?

Compare-and-swap (CAS)

Assume that the shared variable `value` is initialized to zero.

```
Thread 1:
value = 1;
print "(1, " + value + ")";

Thread 2:
old = CAS(value, 0, 2);
print "(2, " + old + ")";
```

Question

What is the output produced and the final value of `value`?

Answer

The output is either "(1, 1) (2, 1)" or "(2, 0) (1, 1)" and the final value is one.

CAS

We use the CAS operation.

Question

How can we implement the push operation?

CAS

We use the CAS operation.

Question

How can we implement the push operation?

Answer

```
push(e):
new = node with element e;
do
  temp = top;
  new.next = temp;
while (CAS(top, temp, new) != temp);
```

CAS

We use the CAS operation.

Question

How can we implement the pop operation?

CAS

We use the CAS operation.

Question

How can we implement the pop operation?

Answer

```
pop():
do
  temp = top;
  if (temp is undefined)
    return EMPTY
while (CAS(top, temp, temp.next) != temp);
return element of temp;
```

java.util.concurrent.atomic

The Java package `java.util.concurrent.atomic` contains classes that support lock-free thread-safe programming on single variables.

AtomicReference⟨V⟩

Objects of type `AtomicReference<V>` contain a value of type V that may be updated atomically.

The class contains the method

```
public final boolean compareAndSet(V expect, V update)
```

It atomically sets the value to `update` if the current value of the object == `expect`. It returns true if the update is successful, and false otherwise.

Node⟨T⟩

```
public class Node<T> {
  private final T data;
  private Node<T> next;

  public Node(T data, Node<T> next) {
    this.data = data;
    this.next = next;
  }
  ...
}
```

AtomicReference⟨V⟩

Problem

Implement a Stack by means of `AtomicReference<V>`.

Stack

```
public class Stack<T> {
  private final AtomicReference<Node<T>> top;

  public Stack() {
    super();
    this.top = new AtomicReference<Node<T>>();
  }
  ...
}
```

Stack

```
public T pop() throws Exception {
  Node<T> node;
  do {
    node = this.top.get();
    if (node == null) {
      throw new Exception();
    }
  }
  while (!this.top.compareAndSet(node, node.getNext()));
  return node.getData();
}
```

Stack

```
public void push(T data) {
  Node<T> node = new Node<T>(data, null);
  do {
    node.setNext(this.top.get());
  }
  while (!this.top.compareAndSet(node.getNext(), node));
}
```

Data race?

Given an empty stack, one thread pushes 1 onto the stack and another thread pops an element from the stack.

Question

Is there a data race?

Data race?

Given an empty stack, one thread pushes 1 onto the stack and another thread pops an element from the stack.

Question

Is there a data race?

Question

According to JPF

```
================================ results
no errors detected
```

Data race?

The AtomicReference class contains the following.

```
import sun.misc.Unsafe;

private static final Unsafe unsafe = Unsafe.getUnsafe();

public final boolean compareAndSet(V expect, V update) {
  return unsafe.compareAndSwapObject(this, valueOffset,
                                     expect, update);
}
```

In JPF the class sun.misc.Unsafe is handled by the native peer class JPF_sun_misc_Unsafe. Hence, the code is executed, not model checked. Therefore, no data race is detected.