Programming Paradigms

Concurrency (Part 1)

Prof. Dr. Michael Pradel

Software Lab, University of Stuttgart

Summer 2020

Wake-up Exercise

What does this Java code print?

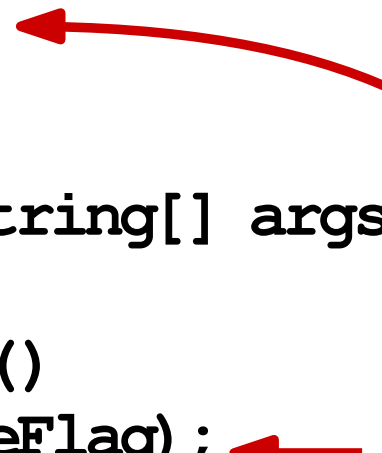
```
class Warmup {
  static boolean flag = false;
  static void raiseFlag() {
    flag = true;
  }
  public static void main(String[] args)
      throws Exception {
    ForkJoinPool.commonPool()
      .execute(Warmup::raiseFlag);
    while (!flag) {};
    System.out.println(flag);
  }
}
```

Please vote via Ilias.

Wake-up Exercise

What does this Java code print?

```
class Warmup {
  static boolean flag = false;
  static void raiseFlag() {
    flag = true;
  }
  public static void main(String[] args)
      throws Exception {
    ForkJoinPool.commonPool()
      .execute(Warmup::raiseFlag);
    while (!flag) {};
    System.out.println(flag);
  }
}
```



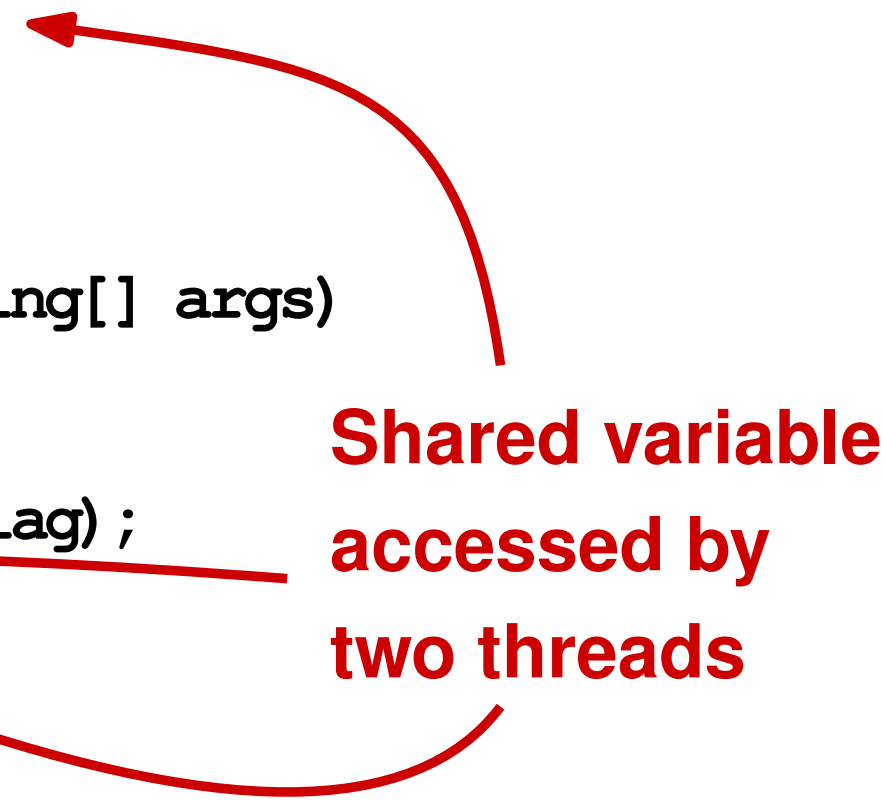
`raiseFlag`: executed in concurrent thread

Please vote via Ilias.

Wake-up Exercise

What does this Java code print?

```
class Warmup {
  static boolean flag = false;
  static void raiseFlag() {
    flag = true;
  }
  public static void main(String[] args)
      throws Exception {
    ForkJoinPool.commonPool()
      .execute(Warmup::raiseFlag);
    while (!flag) {};
    System.out.println(flag);
  }
}
```

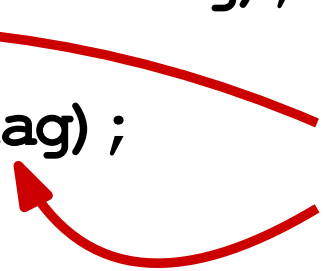


Please vote via Ilias.

Wake-up Exercise

What does this Java code print?

```
class Warmup {
  static boolean flag = false;
  static void raiseFlag() {
    flag = true;
  }
  public static void main(String[] args)
      throws Exception {
    ForkJoinPool.commonPool()
      .execute(Warmup::raiseFlag);
    while (!flag) {};
    System.out.println(flag);
  }
}
```



Problem: No synchronization. Hence, main thread may read old value

Please vote via Ilias.

Wake-up Exercise

What does this Java code print?

```
class Warmup {
  static boolean flag = false;
  static void raiseFlag() {
    flag = true;
  }
  public static void main(String[] args)
      throws Exception {
    ForkJoinPool.commonPool()
      .execute(Warmup::raiseFlag);
    while (!flag) {};
    System.out.println(flag);
  }
}
```

Code may hang forever,

print `true`, or print `false`!

Please vote via Ilias.

Overview

- Introduction
- Concurrent Programming Fundamentals
- Implementing Synchronization
- Language-level Constructs

Motivation

Why do we care about concurrency?

- To capture the logical structure of a problem
  - Inheritently concurrent problems, e.g., server handling multiple requests
- To exploit parallel hardware for speed
  - Since around 2005: Multi-core processors are the norm
- To cope with physical distribution
  - Local or global groups of interacting machines

Terminology

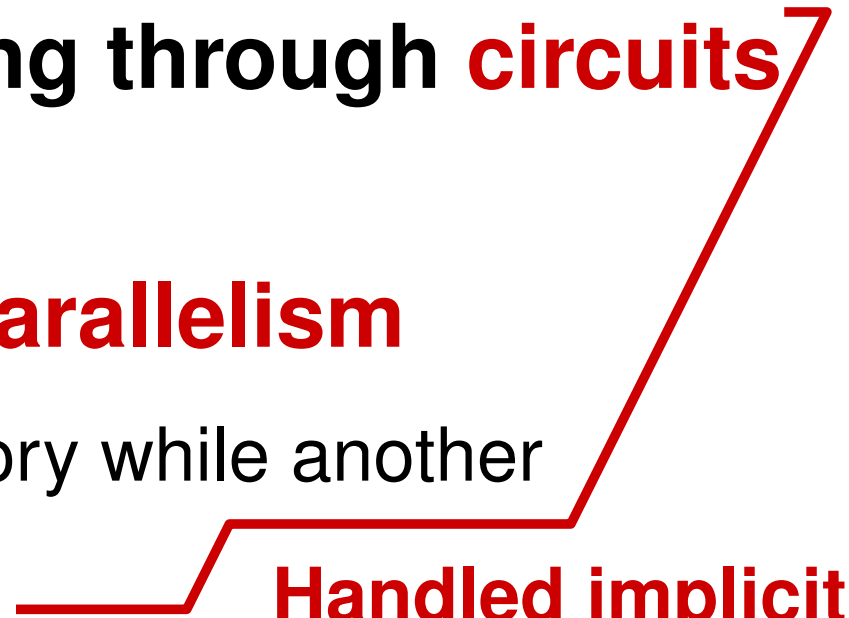
- **Concurrent**
  - Two or more running tasks whose execution may be at some unpredictable point
- **Parallel**
  - Two or more tasks are actively executing at the same time
  - Requires multiple processor cores
- **Distributed**
  - Physically separated processors

Levels of Parallelism

- **Signals propagating through circuits and gates**
- **Instruction-level parallelism**
  - E.g., load from memory while another instruction executes
- **Vector parallelism**
  - E.g., GPUs execute a single instruction on a vector of data
- **Thread-level parallelism**

Levels of Parallelism

- **Signals propagating through circuits and gates**
- **Instruction-level parallelism**
  - E.g., load from memory while another instruction executes
- **Vector parallelism**
  - E.g., GPUs execute a single instruction on a vector of data
- **Thread-level parallelism**



Levels of Parallelism

- **Signals propagating through circuits and gates**
- **Instruction-level parallelism**
  - E.g., load from memory while another instruction executes
- **Vector parallelism**
  - E.g., GPUs execute a single instruction on a vector of data
- **Thread-level parallelism**

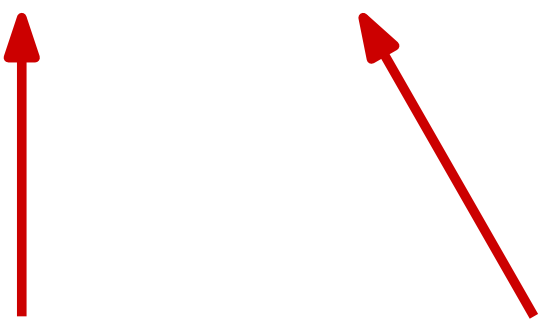
Specified by programmer in PL

Example: Independent Tasks

```
// Task Parallel Library in C#
Parallel.For(0, 100, i => {
  A[i] = foo(A[i]);
});
```

Example: Independent Tasks

```
// Task Parallel Library in C#
Parallel.For(0, 100, i => {
  A[i] = foo(A[i]);
});
```

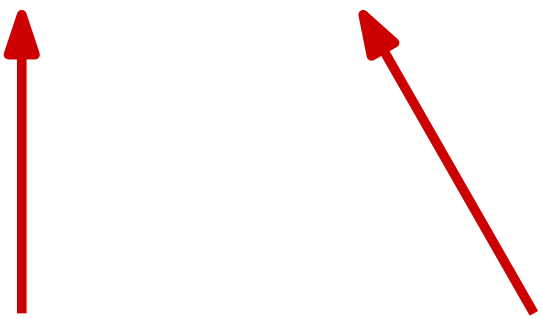


Array of data

Function that updates each element independently

Example: Independent Tasks

```
// Task Parallel Library in C#
Parallel.For(0, 100, i => {
  A[i] = foo(A[i]);
});
```



Array of data

Function that updates each element independently

- No need to synchronize tasks
- Uses as many cores as possible (up to 100)

Example: Dependent Tasks

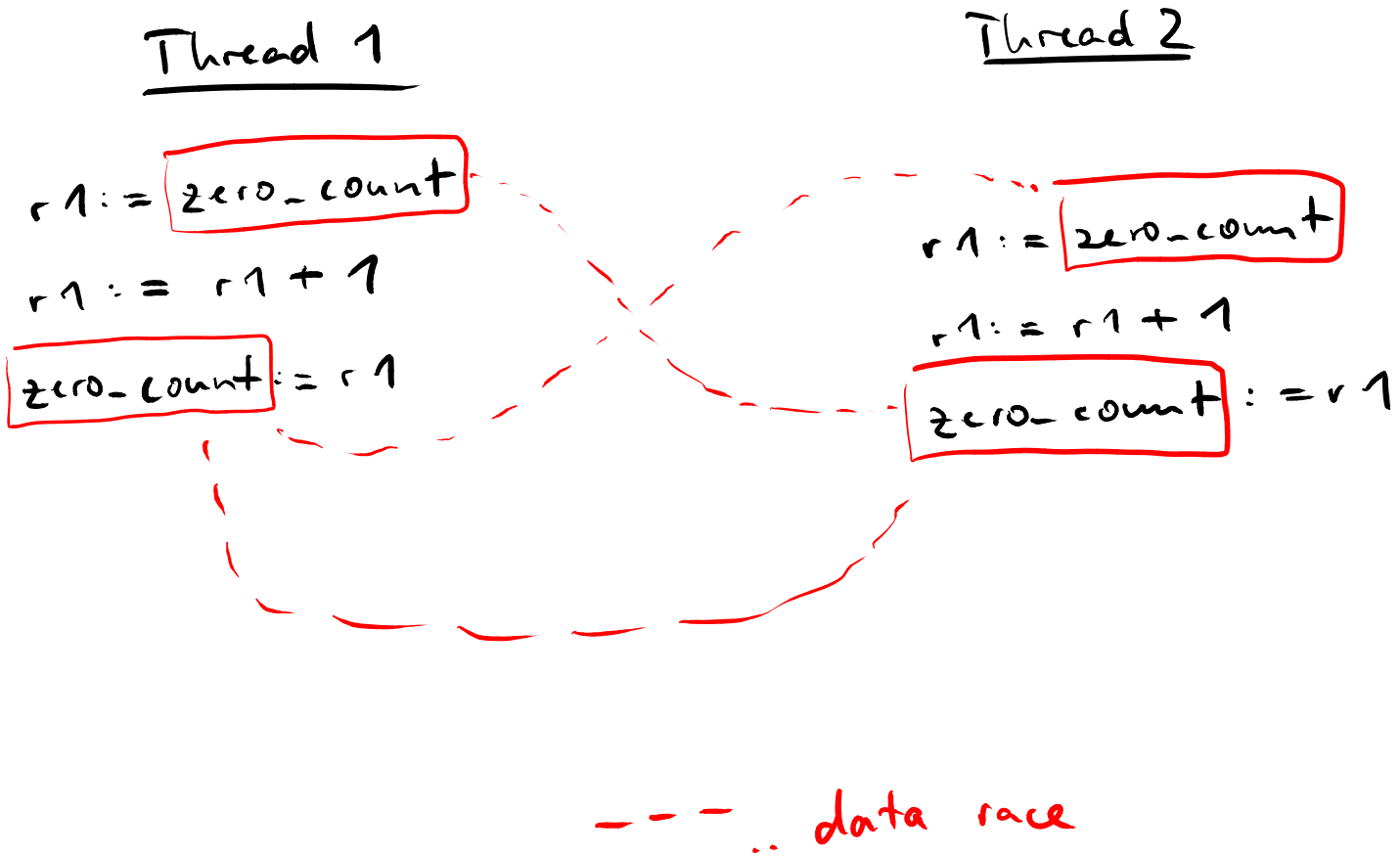
```
// As before, but foo now is:
int zero_count;
public static int foo(int n) {
   int rtn = n - 1;
   if (rtn == 0) zero_count++;
   return rtn;
}
```

Example: Dependent Tasks

```
// As before, but foo now is:
int zero_count;
public static int foo(int n) {
   int rtn = n - 1;
   if (rtn == 0) zero_count++;
   return rtn;
}
```

Count how many zeros written to the array

Data Race


Thread 1
r1 := zero_count
r1 := r1 + 1
zero_count := r1
Thread 2
r1 := zero_count
r1 := r1 + 1
zero_count := r1
--- .. data race

Data Races

- **Definition of data race**
  - Two accesses to the same shared memory location
  - At least one access is a write
  - Ordering of accesses is non-deterministic