

CMSC 132 Quiz 4 Worksheet

The next quiz of the course will be on Friday, July 27. The following list provides more information about the quiz:

- The quiz will be a written quiz (no computer).
- Closed book, closed notes quiz.
- Answers must be neat and legible. We recommend that you use pencil and eraser.

The following exercises cover the material to be included in this quiz. Solutions to these exercises will not be provided, but you are welcome to discuss your solutions with TAs and instructors during office hours.

Thread/Synchronization

1. What are two advantages to multi-threading?
2. What are two disadvantages to using multi-threading?
3. What are two ways to create threads in Java?
4. What is a daemon thread?
5. What is a deadlock? How can you avoid it?
6. What is a data race? How can you avoid it?
7. Give an example of a Java code with a data race.
 - a. Eliminate the data race using synchronized methods, e.g., synchronized foo() { ... }
 - b. Eliminate the data race using synchronized objects, e.g., synchronized(bar) { ... }
8. Modify the following class so we can create threads that print messages. For the modified class, provide a main method that creates and starts two threads, one printing the message “Testudo” and the other the message “Terps”. A third thread displaying “UMCP” will be created and started only after the previous two threads and the main thread have finished.

```
public class PrtMessage {
    private String message;

    public PrtMessage(String message) {
        this.message = message;
    }

    public void print() {
        for (int i=0; i<50; i++)
            System.out.println(message);
    }
}
```

9. The following class implements a queue.

```
public class MyQueue<E> {
    private ArrayList<E> list = new ArrayList<E>();

    public boolean isEmpty() {
        synchronized(this) {
            if (list.size() == 0)
                return true;
            return false;
        }
    }
}
```

```

public E getFirst() {
    synchronized(this) {
        return list.remove(0); // removes first element and shift
                                // elements to the right
    }
}

public void offer(E data) {
    synchronized(this) {
        list.add(data);
    }
}

/** If queue not empty, remove value from queue and return it.
 * Otherwise, if queue is empty, return null */
public E dequeue() {
    if (!isEmpty())
        return getFirst();
    return null;
}
}

```

- i. Describe a possible scenario where the dequeue method will not work as documented when the dequeue is accessed by multiple threads.
- ii. Modify the **offer** and **dequeue** methods in order to allow the dequeue method to wait until it can return a value when the queue is empty it (as opposed to returning null).