Chapter 32 Check Point Questions

Section 32.2

▼32.2.1

(1) Why is multithreading needed? (2) How can multiple threads run simultaneously in a single-processor system?

(1) Multithreading can make your program more responsive and interactive, and enhance the performance. Multithreading is needed in many situations, such as animation and client/server computing.
(2) Because most of time the CPU is idle--for example, the CPU is doing nothing while the user enters data--it is practical for multiple threads to share the CPU time in single-processor systems.

Hide Answer

▼32.2.2

What is a runnable object? What is a thread?

An instance of Runnable is a runnable object. A thread is wrapper object for a runnable object for executing a runnable task.

Hide Answer

Section 32.3

▼32.3.1

How do you define a task class? How do you create a thread for a task?

You can create a task classes by implementing the Runnable interface and create a thread for a task using the constructor new Thread(task).

Hide Answer

▼32.3.2

What would happen if you replace the start() method with the run() method in lines 14-16 in Listing 32.1?

If you replace the start() method by the run() method in Lines 11-13 in Example 19.1, the run() method are executed in sequence. The threads are not executed concurrently.

Hide Answer

▼32.3.3

What is wrong in the following two programs? Correct the errors.

```
(a)
public class Test implements Runnable {
  public static void main(String[] args) {
    new Test();
  }
  public Test() {
    Test task = new Test();
    new Thread(task).start();
  }

  public void run() {
    System.out.println("test");
  }
}

(b)
public class Test implements Runnable {
  public static void main(String[] args) {
    new Test();
  }

  public Test() {
    Thread t = new Thread(this);
    t.start();
    t.start();
  }

  public void run() {
    System.out.println("test");
  }
}
```

(a) new Test() is recursively called inside the constructor. To fix it, delete the highlighted line and use new Thread(this).start().
(b) An illegal java.lang.IllegalThreadStateException may be thrown because you just started thread and thread might have not yet finished before you start it again. To fix it, delete one t.start().

Hide Answer

Section 32.4

▼32.4.1

Which of the following methods are instance methods in java.lang.Thread? Which method may throw an InterruptedException? Which of them are deprecated in Java?
run, start, stop, suspend, resume, sleep, interrupt, yield, join

All of the methods shown in Figure 30.4 except yield() and sleep(long) are instance methods. sleep() and join() may throw an InterruptedException. The methods stop(), suspend() and resume() are deprecated in JDK 1.2.

Hide Answer

▼32.4.2

If a loop contains a method that throws an InterruptedException, why should the loop be placed inside a try-catch block?

If the loop is outside the try-catch block, the thread may continue to execute even though it is being interrupted.

Hide Answer

▼32.4.3

How do you set a priority for a thread? What is the default priority?

You use the setPriority() method to set the priority for a thread. The default priority of the thread is Thread.NORM_PRIORITY (5).

Hide Answer

Section 32.5

▼32.5.1
What causes the text to flash?

You see the text flashing, because the program displays a text on the label and then display nothing on the label. This alternates to cause the flashing effect.

Hide Answer

▼32.5.2
Is an instance of FlashText a runnable object?

No. Because it does not implement the Runnable interface.

Hide Answer

▼32.5.3
What is the purpose of using Platform.runLater?

Invoking Platform.runLater(Runnable r) tells the system to run a task in the JavaFX application thread.

Hide Answer

▼32.5.4
Can you replace the code in lines 27-32 using the following code?

```
Platform.runLater(e -> lblText.setText(text));
```

Yes.

Hide Answer

▼32.5.5
What happens if line 34 (Thread.sleep(200)) is not used?

The while loop will get all the time. It starves all other tasks.

Hide Answer

▼32.5.6
There is an issue in Listing 16.9 ListViewDemo. If you press the CTRL key and select Canada, Demark, and China in this order, you will get an ArrayIndexOutBoundsException. What is the reason for this error and how do you fix it? (Thanks to Henri Heimonen of Finland for contributing this question 04/30/2016).

The event handler for list item selection is executed on a separate thread that is different from the JavaFX GUI thread. To ensure that the handler is executed after the GUI action is completed, execute the handler on the JavaFX GUI thread using the following code:

```
lv.getSelectionModel().selectedItemProperty().addListener(ov -> {
  Platform.runLater(() -> { // Run from JavaFX GUI
    imagePane.getChildren().clear();
    for (Integer i : lv.getSelectionModel().getSelectedIndices()) {
      imagePane.getChildren().add(ImageViews[i]);
    }
  });
});
```

Hide Answer

Section 32.6

▼32.6.1

What are the benefits of using a thread pool?

You can create a thread for each task. This approach is convenient for a single task execution, but it is not efficient for a large number of tasks, because you have to create a thread for each task. Starting a new thread for each task could limit throughput and cause poor performance. A thread pool is ideal to manage the number of tasks executing concurrently.

Hide Answer

▼32.6.2

How do you create a thread pool with three fixed threads? How do you submit a task to a thread pool? How do you know that all the tasks are finished?

To create a thread pool with three threads, use

```
ExecutorService executor = Executors.newFixedThreadPool(3);
```

To submit a task, use

```
executor.execute(task);
```

To check whether all tasks in a the pool are finished, invoke isTerminated() method.

Hide Answer

Section 32.7

▼32.7.1

Give some examples of possible resource corruption when running multiple threads. How do you synchronize conflicting threads?

See the section "Synchronization" for examples and solutions.

Hide Answer

▼32.7.2

Suppose you place the statement in line 26 of Listing 32.4 inside a synchronized block to avoid race conditions, as follows:

```
synchronized (this) {
  account.deposit(1);
}
```

Will it work?

No. synchronized (this) acquires a lock on a thread (an instance of AddAPennyThread). Each thread still can access the object bank concurrently. To fix the problem, acquire the lock on bank using synchronized (bank).

Hide Answer

Section 32.8

▼**32.8.1**

How do you create a lock object? How do you acquire a lock and release a lock?

To create a lock, use the new ReentrantLock(). To acquire the lock, invoke its lock() method and to release it, invoke its unlock() method.

Hide Answer

Section 32.9

▼**32.9.1**

How do you create a condition on a lock? What are the await(), signal(), and signalAll() methods for?

A condition on a lock can be created using lock.newCondition(). The await() method causes the current thread to wait until the condition is signaled. The signal() method wakes up one waiting thread, and the signalAll() method wakes up all waiting threads.

Hide Answer

▼**32.9.2**

What would happen if the while loop in line 58 of Listing 32.6 was changed to an if statement?

```
if (balance < amount)
```

When a thread notify a waiting thread, you cannot assume the balance >= amount for the waiting thread. The condition (balance < amount) may be still true when the thread is awakened.

Hide Answer

▼32.9.3
Why does the following class have a syntax error?

```
public class Test implements Runnable {
  public static void main(String[] args) {
    new Test();
  }

  public Test() throws InterruptedException {
    Thread thread = new Thread(this);
    thread.sleep(1000);
  }

  public synchronized void run() {
  }
}
```

To override the init() method defined in the Applet class, you have to use the exact signature. The init() method does not claim throwing exceptions.

Hide Answer

▼32.9.4
What is a possible cause for IllegalMonitorStateException?

If you invoke the methods on a condition without first acquiring a lock for the condition, an IllegalMonitorStateException would be thrown.

Hide Answer

▼32.9.5
Can the wait(), notify(), and notifyAll() be invoked from any object? What is the purpose of these methods?

Yes. These methods are defined in the Object class and they are used for thread communication.

Hide Answer

▼32.9.6
What is wrong in the following code?

```
synchronized (object1) {
  try {
    while (!condition) object2.wait();
  }
  catch (InterruptedException ex) {
  }
}
```

object1 and object2 are not the same. A lock must be acquired on the receiving object of the wait(), notify(), and notifyAll() methods before invoking these methods.

Hide Answer

Section 32.10

▼32.10.1
Can the read and write methods in the Buffer class be executed concurrently?

No.

Hide Answer

▼**32.10.2**
When invoking the read method, what happens if the queue is empty?

It will wait for a signal for the queue to be not empty.

Hide Answer

▼**32.10.3**
When invoking the write method, what happens if the queue is full?

It will wait for a signal for the queue to be not full.

Hide Answer

Section 32.11

▼**32.11.1**
What is a blocking queue? What blocking queues are supported in Java?

The ArrayBlockingQueue, LinkedBlockingQueue, and PriorityBlockingQueue are supported in JDK 1.5. All are concrete classes of the BlockingQueue interface.

Hide Answer

▼**32.11.2**
What method do you use to add an element to an ArrayBlockingQueue? What happens if the queue is full?

Use put(element) method to add an element to the queue. If the queue is full, the thread is blocked.

Hide Answer

▼**32.11.3**
What method do you use to retrieve an element from an ArrayBlockingQueue? What happens if the queue is empty?

Use take() method to retrieve an element from the queue. If the queue is empty, the thread is blocked.

Hide Answer

Section 32.12

▼**32.12.1**
What are the similarities and differences between a lock and a semaphore?

Lock and semaphore can both be used to restrict access to a shared resource. Using a lock on a resource ensures only one thread can access it. Using a semaphore on a resource allows one or more specified number of threads to access a resource.

Hide Answer

▼**32.12.2**
How do you create a semaphore that allows three concurrent threads? How do you acquire a

semaphore? How do you release a semaphore?

Use new Semaphore(numberOfPermits) to create a semaphore. Invoking aquire() to get a semaphore and invoking release() to release a semaphore.

Hide Answer

Section 32.13

▼32.13.1

What is a deadlock? How can you avoid deadlock?

Deadlock occurs in the case that two or more threads acquire locks on multiple objects and each has the lock on one object and is waiting for the lock on the other object. The resource ordering technique can be used to avoid deadlock.

Hide Answer

Section 32.14

▼32.14.1

What is a thread state? Describe the states for a thread.

See the text.

Hide Answer

Section 32.15

▼32.15.1

What is a synchronized collection? Is ArrayList synchronized? How do you make it synchronized?

A synchronized collection is thread-safe, i.e., it can be accessed by multiple threads concurrently without being corrupted. ArrayList is not thread-safe. There are several ways to make it safe. You may obtain a lock before accessing it, or use the Collections.synchronizedList(list) to return a synchronized list.

Hide Answer

▼32.15.2

Explain why an iterator is fail-fast.

An iterator is fail-fast. This means that if you are using an iterator to traverse a collection while the underlying collection is being modified by another thread, then the iterator will immediately fail by throwing java.util.ConcurrentModificationException.

Hide Answer

Section 32.16

▼32.16.1

How do you define a ForkJoinTask? What are the differences between RecursiveAction and RecursiveTask?

To define a ForkJoinTask, define a class that extends RecursiveAction or RecursiveTask. RecursiveAction is for a task that doesn't return a value and RecursiveTask is for a task that returns a value.

Hide Answer

▼32.16.2
How do you tell the system to execute a task?

Invoke the invoke method on a ForkJoinTask object to execute a task.

Hide Answer

▼32.16.3
What method can you use to test if a task has been completed?

Invoke the isDone method to test if the task is completed.

Hide Answer

▼32.16.4
How do you create a ForkJoinPool? How do you place a task into a ForkJoinPool?

You can create a fork join pool using new ForkJoinPool(). To place a task to the pool, use the invoke method to add a task.

Hide Answer