問 11　次の Java プログラムの説明及びプログラムを読んで，設問 1，2 に答えよ。
（Java プログラムで使用する API の説明は，この冊子の末尾を参照してください。）

〔プログラムの説明〕

スレッドを利用してタイマ処理を行うプログラムである。タイマ処理の基本機能は，秒単位で指定した時間（以下，遅延時間という）が経過したときに，インタフェースで指定したメソッドを呼び出すことである。次のクラス及びインタフェースから成る。

(1) クラス Timer は，タイマ処理の機能を定義し実装する。

① クラスメソッド createTimer：クラス Timer のインスタンス（以下，タイマという）を生成し，直ちにタイマ処理を開始する。引数で，タイマの名前，インタフェース TimerAction を実装するインスタンス及び遅延時間を指定する。

② メソッド getName：タイマの名前を返す。

③ メソッド cancel：タイマ処理をキャンセルする。

④ メソッド close：タイマ処理が終了するまで待機する。

⑤ 内部クラス Worker：タイマ処理を実装し，そのインスタンスは，スレッドとして実行される。実行スレッドは，スレッド名 Timer-*n*（*n* は 0 から始まるスレッド生成のシーケンス番号）が与えられ，次の処理を行う。

(a) 現在時刻に，タイマの生成時に指定された遅延時間を加えて，タイマの終了時刻を設定する。

(b) 実行開始時に，タイマの生成時に指定された TimerAction のインスタンスに開始の事象を通知する。

(c) 開始通知後，スレッドは終了時刻になるまで休止する。

(d) 終了時刻になったとき，終了の事象を TimerAction のインスタンスに通知し，タイマ処理の実行を終了する。

(e) タイマ処理がキャンセルされた場合は，終了時刻になるのを待たずにキャンセルの事象を TimerAction のインスタンスに通知し，タイマ処理の実行を終了する。

ここで，フィールド canceled は，複数スレッドから非同期にアクセスされるので volatile 修飾子を付けて宣言しておく。

(2) インタフェース TimerAction は，タイマ処理で事象が発生したときにタイマから事象の通知を受けるメソッドを定義する。タイマを利用するプログラムは，このインタフェースの全メソッドを実装しなければならない。各メソッドの引数は，事象が発生した Timer のインスタンス及び事象発生の時刻である。

① メソッド onStart：タイマ処理の開始時に呼び出される。

② メソッド onAlarm：タイマ生成時に指定した遅延時間が経過したときに呼び出される。

③ メソッド onCancel：タイマ処理がキャンセルされたときに呼び出される。

クラス TimerTest は，タイマ処理のテストプログラムである。メソッド main は，TimerTest のインスタンスを生成し，メソッド test を実行する。test は，二つのタイマを生成し，終了を待つ。メソッド onAlarm が，shortTimer を引数として呼び出された場合，longTimer のタイマ処理をキャンセルする。

このテストを実行したところ，図1の結果が得られた。プログラムの実行速度は十分に速いものとし，各メソッドの処理時間は秒単位の時刻の測定に影響がないものとする。


```
long timer: onStart at Mon Oct 01 21:01:40 JST 2012
short timer: onStart at Mon Oct 01 21:01:40 JST 2012
short timer: onAlarm at Mon Oct 01 21:01:42 JST 2012
long timer: onCancel at Mon Oct 01 21:01:42 JST 2012
```

図1　テストプログラムの実行結果

〔プログラム1〕

```
public interface TimerAction {
   public void onStart(Timer timer, long instant);
   public void onAlarm(Timer timer, long instant);
   public void onCancel(Timer timer, long instant);
}
```

〔プログラム2〕

```
public class Timer {
   private static int sequence;
   private String name;
   private Worker worker;
   private Thread thread;
   private TimerAction timerAction;

   private Timer(String name, TimerAction timerAction,
                 int delay) {
      this.name = name;
      this.timerAction = timerAction;
      worker = new Worker(delay);
      thread = new Thread(worker, getThreadSequenceName());
   }

   synchronized private static String getThreadSequenceName() {
      return "Timer-" + sequence++;
   }

   private void start() { thread.start(); }

   public static Timer createTimer(String name,
                                   TimerAction action, int delay) {
      Timer timer = new Timer(name, action, delay);
      timer.start();
      return timer;
   }

   public String getName() { return name; }

   public void cancel() { worker.cancel(); }

   public void close() throws InterruptedException {
      thread.join();
   }

   private class Worker implements Runnable {
      private final long endAt;
      private volatile boolean canceled;

      private Worker(int delay) {
```

```
            endAt = currentTime() + delay * 1000;
        }

        public void run() {
            timerAction.onStart(Timer.this, currentTime());
            long delta;
            while ( [ a ] ) {
                try {
                    Thread.sleep(delta);
                } catch (InterruptedException e) {
                    if (canceled) {
                        [ b ];
                    }
                }
            }
            if (canceled) {
                timerAction.onCancel(Timer.this, currentTime());
            } else {
                timerAction.onAlarm(Timer.this, currentTime());
            }
        }

        private void cancel() {
            canceled = true;
            Timer.this.thread.interrupt();
        }

        private long currentTime() {
            return System.currentTimeMillis();
        }
    }
}
```

〔プログラム3〕

```
import java.util.Date;

public class TimerTest [ c ] TimerAction {
   Timer longTimer, shortTimer;

   private void test() [ d ] InterruptedException {
      longTimer = Timer.createTimer("long timer", [ e ], 4);
      shortTimer = Timer.createTimer("short timer", [ e ], 2);
      shortTimer.close();
      longTimer.close();
   }

   private void log(String msg, Timer timer, long instant) {
      System.out.println(timer.getName() + ": " + msg
                         + " at " + new Date(instant));
   }

   public void onStart(Timer timer, long instant) {
      log("onStart", timer, instant);
   }

   public void onAlarm(Timer timer, long instant) {
      log("onAlarm", timer, instant);
      if (timer == shortTimer) {
         [ f ];
      }
   }

   public void onCancel(Timer timer, long instant) {
      log("onCancel", timer, instant);
   }

   public static void main(String[] args) [ d ]
                                    InterruptedException {
      new TimerTest().test();
   }
}
```

設問1　プログラム中の［　　］に入れる正しい答えを，解答群の中から選べ。

aに関する解答群

ア　(delta = endAt + currentTime()) < 0

イ　(delta = endAt + currentTime()) > 0

ウ　(delta = endAt - currentTime()) < 0

エ　(delta = endAt - currentTime()) > 0

bに関する解答群

ア　break	イ　canceled = false	ウ　continue
エ　delta = 0	オ　endAt = 0	カ　return

c，dに関する解答群

ア　expands	イ　extends	ウ　implements
エ　subclasses	オ　throw	カ　throws

eに関する解答群

ア　longTimer	イ　new Runnable()	ウ　new Timer()
エ　new TimerAction()	オ　shortTimer	カ　this

fに関する解答群

ア　longTimer.cancel()	イ　onCancel(longTimer, instant)
ウ　onCancel(shortTimer, instant)	エ　onCancel(timer, instant)
オ　shortTimer.cancel()	カ　timer.cancel()

Java

設問2　クラス TimerTest に，メソッド log をオーバロードする次のメソッドを追加した。このメソッドは，タイマの名前の代わりにスレッド名を表示する。

```
private void log(String msg, long instant) {
   System.out.println(Thread.currentThread().getName()
           + ": " + msg + " at " + new Date(instant));
}
```

メソッド onAlarm 及び onCancel で，このオーバロードしたメソッド log を呼び出すように変更したところ，図2の実行結果が得られた。図2の3行目及び4行目の［　　］に入れる文字列の組合せとして正しい答えを，解答群の中から選べ。ここで，メソッド main を実行するスレッドの名前は main とする。また，［a］～［f］には正しい答えが入っているものとする。

```
long timer: onStart at Mon Oct 01 23:45:15 JST 2012
short timer: onStart at Mon Oct 01 23:45:15 JST 2012
[g1]: onAlarm at Mon Oct 01 23:45:17 JST 2012
[g2]: onCancel at Mon Oct 01 23:45:17 JST 2012
```

図2　TimerTest 変更後の実行結果

解答群

	g1	g2
ア	main	main
イ	main	Timer-0
ウ	main	Timer-1
エ	Timer-0	main
オ	Timer-0	Timer-1
カ	Timer-1	main
キ	Timer-1	Timer-0