選択した問題は，選択欄の(選)をマークしてください。マークがない場合は，採点されません。

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〕

スマートフォンやタブレット端末といった携帯端末に通知メッセージを配信するシステム（以下，通知システムという）を模したプログラムである。この通知システムは，メール着信を通知するメッセージを非同期で携帯端末に配信する。このプログラムでは，通知メッセージを配信する処理及び各携帯端末の処理を，それぞれ独立したスレッドとして実行する。

このプログラムは，次のインタフェース及びクラスから成る。ここで，各コンストラクタ及びメソッドには，正しい引数が与えられるものとする。

(1) インタフェース NotificationListener は，通知メッセージを受け取るためのメソッドを定義する。以下，NotificationListener のインスタンスをリスナという。
　① メソッド onNotificationReceived は，通知メッセージを受信したときに呼び出される。受信した通知メッセージは，引数の文字列のリストで与えられる。
(2) クラス MobileDevice は，携帯端末を表す。
　① コンストラクタは，引数で指定された携帯端末名及びリスナをもつ携帯端末を生成する。
　② メソッド getListener はリスナを，getName は携帯端末名を返す。
(3) クラス Notifier は，携帯端末の管理や通知メッセージの配信などを行う。Notifier のインスタンスは，シングルトン（Java 仮想計算機内で唯一の存在）である。
　① メソッド register は，引数で指定された利用者名とその携帯端末名を登録する。指定された利用者名に対応する携帯端末名が既に登録されている場合は，その利用者名に対応する携帯端末名として追加登録する。

② メソッド send は，引数で指定された利用者名で登録されている各携帯端末に，引数で指定された文字列を通知メッセージとして配信する。携帯端末に対して未配信の通知メッセージがある場合，引数のメッセージを未配信のメッセージリストに追加する。

③ メソッド loopForMessages は，引数で指定された携帯端末に対して，通知メッセージがあれば携帯端末のリスナに通知し，なければ通知メッセージを受け取れる状態（以下，待ち受け状態という）にする。この処理を，通知システムが停止されるまで繰り返す。

④ メソッド shutdown は，通知システムを停止する。未配信の全メッセージ，全利用者名及び全携帯端末名の登録情報を削除し，登録されている全携帯端末の待ち受け状態を解除する。

(4) クラス Tester は，プログラム 1～3 をテストする。

① メソッド main は，利用者名 Taro の携帯端末名 phone 及び tablet を通知システムに登録して，Taro にメッセージを送信する。その後，通知システムを停止する。

② メソッド createUserMobileDevice は，利用者名とその携帯端末名を登録して，通知メッセージを受信できる状態にする処理を，新しく生成したスレッドで実行する。

図 1 は，クラス Tester のメソッド main を実行して得られた出力の例である。ここで，プログラムは，スレッドの実行速度及び事象発生に対する応答が十分速いシステムで実行されるものとする。また，スレッドのスケジューリングによって，各行の出力順は異なることがある。

```
phone: [You have a message.]
tablet: [You have a message.]
Terminating Taro's tablet
Terminating Taro's phone
```

図 1　メソッド main の実行結果の例

〔プログラム 1〕

```
import java.util.List;

public interface NotificationListener {
    void onNotificationReceived(List<String> messageList);
}
```

〔プログラム 2〕

```
public final class MobileDevice {
    private final String name;
    private final NotificationListener listener;

    public MobileDevice(String name, NotificationListener listener) {
        this.name = name;
        this.listener = listener;
    }

    public NotificationListener getListener() { return listener; }

    public String getName() { return name; }
}
```

〔プログラム 3〕

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public final class Notifier {
    private static final Notifier INSTANCE = new Notifier();

    private final Object lock = new Object();
    // 利用者ごとに携帯端末を管理
    private final Map<String, List<MobileDevice>> userMobileDevices
            = new HashMap<>();
    // 携帯端末ごとに通知メッセージを保持
    private final Map<MobileDevice, List<String>> messagesToDeliver
            = new HashMap<>();
    private volatile boolean active = true;

    public static Notifier getInstance() { return [ a ]; }
```

```
private Notifier() { }

public void register(String user, MobileDevice device) {
   synchronized (lock) {
      List<MobileDevice> devices = userMobileDevices.get(user);
      if (devices == null) {
         devices = new ArrayList<>();
         userMobileDevices.put(    b    );
      }
      devices.add(device);
   }
}

public void send(String user, String message) {
   List<MobileDevice> devices = new ArrayList<>();
   synchronized (lock) {
      if (userMobileDevices.containsKey(user)) {
         for (MobileDevice device : userMobileDevices.get(user)) {
            List<String> messageList = messagesToDeliver.get(device);
            if (messageList == null) {
               messageList = new ArrayList<>();
               messagesToDeliver.put(    c    );
            }
            messageList.add(message);
            devices.add(device);
         }
      }
   }
   for (MobileDevice device : devices) {
      synchronized (device) {
         // 通知メッセージがあることを待ち受け状態のスレッドに通知
         device.notifyAll();
      }
   }
}

public void loopForMessages(MobileDevice device) {
   while (active) {
      List<String> messageList;
      synchronized (lock) {
         messageList = messagesToDeliver.remove(device);
      }
```

```
            if (messageList != null) {
                device.getListener().onNotificationReceived(messageList);
            }
            synchronized (device) {
                try {
                    // 通知メッセージが到着するかタイムアウトするまで待つ
                    device.wait(3000L);
                } catch (InterruptedException e) {
                    break;
                }
            }
        }
    }

    public void shutdown() {
        active = false;
        List<MobileDevice> devices = new ArrayList<>();
        synchronized (lock) {
            messagesToDeliver.clear();
            for (String user : userMobileDevices.keySet()) {
                for (MobileDevice device : userMobileDevices.get(user)) {
                    devices.add(device);
                }
            }
            userMobileDevices.clear();
        }
        for (MobileDevice device : devices) {
            synchronized (device) {
                // 待ち受け状態のスレッドに通知
                device.notifyAll();
            }
        }
    }
}
```

〔プログラム4〕

```
public class Tester {
    public static void main(String[] args) [    d    ] InterruptedException {
        createUserMobileDevice("Taro", "phone");
        createUserMobileDevice("Taro", "tablet");
        Notifier notifier = Notifier.getInstance();
        notifier.send("Taro", "You have a message.");
```

```
        Thread.sleep(500L);
        notifier.shutdown();
        /* α */
    }

    private static void createUserMobileDevice(String user, String name) {
        MobileDevice device = new MobileDevice(name, messageList ->
                System.out.println(  e  + ": " + messageList));
        Notifier notifier = Notifier.getInstance();
        notifier.register(user, device);
        new Thread(() -> {
            notifier.loopForMessages(device);
            System.out.printf("Terminating %s's %s%n", user, name);
        }).start();
    }
}
```

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	getInstance()	イ	INSTANCE	ウ	new Notifier()
エ	Notifier()	オ	Notifier.class	カ	this

b，cに関する解答群

ア	device, devices	イ	device, messageList	ウ	device, user
エ	user, device	オ	user, devices	カ	user, messageList

dに関する解答群

ア	extends	イ	implements	ウ	requires
エ	throw	オ	throws	カ	uses

eに関する解答群

ア	device	イ	device.getName()	ウ	name
エ	Tester.this.name	オ	this.name	カ	user

設問2　プログラム4のクラスTesterにおいて，メソッドmainの /* α */ を図2の2行で置き換えて実行したとき，この2行に対するプログラムの動作に関する記述として，正しい答えを，解答群の中から選べ。

```
notifier.send("Taro", "You have 2 messages.");
Thread.sleep(500L);
```

図2　/* α */ と置き換える行

解答群

ア　“phone: [You have 2 messages.]” だけを出力する。

イ　“tablet: [You have 2 messages.]” だけを出力する。

ウ　メソッドloopForMessagesで，NullPointerExceptionが発生する。

エ　メソッドsendで，NullPointerExceptionが発生する。

オ　利用者名Taroが登録されていないので，メソッドsendは何もしないで終了する。

カ　利用者名Taroは登録されているが，利用者名Taroの携帯端末が何も登録されていないので，メソッドsendは何もしないで終了する。