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〕

“すべきこと”（以下，ToDo という）を管理するプログラムである。

(1) クラス ToDo は ToDo を表すクラスであり，コンストラクタで主題，期限，重要度を指定する。期限は，年月日を表す 8 桁又は年月日時分を表す 12 桁の数字から成る文字列（以下，日時という）であり，例えば，2016 年 4 月 16 日を表す文字列は“20160416”，2016 年 4 月 16 日午後 1 時 0 分を表す文字列は“201604161300”である。ここで，日時に誤りはないものとする。

主題，期限，重要度を取得する各メソッドと，状態を設定及び取得するメソッド，ToDo を識別するフィールド id をもつ。

列挙 Priority は ToDo の重要度を表す列挙であり，重要度が低い順に LOW，MIDDLE，HIGH である。

列挙 State は ToDo の状態を表す列挙であり，NOT_YET_STARTED は未着手，STARTED は着手済み，DONE は完了を表す。

(2) クラス ToDoList は ToDo のリストを保持するクラスである。

リスト中に，フィールド id の値が同じ ToDo を複数個含まないことを保証する。

ToDo を追加するメソッド add と，ToDo の更新を行うメソッド update，条件に合う ToDo のリストを返すメソッド select をもつ。

メソッド add の引数に，既にリストに保持されている ToDo を指定したとき，及びメソッド update の引数に，リストにない ToDo を指定したときは何もしない。

メソッド select の引数には，条件を 0 個以上指定できる。条件を指定したときは，全ての条件に合致する ToDo から成るリストを返す。条件を指定しないときは，保持する全ての ToDo から成るリストを返す。

(3) インタフェース Condition は，ToDo を選択する際の条件を表すクラスが実装するインタフェースである。メソッド test は条件に合致するときに true を返す。

(4) クラス ToDoListTester は，テスト用のクラスである。

〔プログラム1〕

```
import java.util.UUID;

public class ToDo {
   public enum Priority { LOW, MIDDLE, HIGH }
   public enum State { NOT_YET_STARTED, STARTED, DONE }

   // 8桁又は12桁の数字から成る文字列と一致する正規表現
   private static final String DEADLINE_PATTERN = "\\d{8}|\\d{12}";

   private final String id;
   private String subject;
   private String deadline;
   private Priority priority;
   private State state;

   private ToDo(String subject, String deadline, Priority priority,
                String id, State state) {
      if (!deadline.matches(DEADLINE_PATTERN)) {
         throw new IllegalArgumentException();
      }
      this.subject = subject;
      this.deadline = deadline;
      this.priority = priority;
      this.id = id;
      this.state = state;
   }

   public ToDo(String subject, String deadline, Priority priority) {
      this(subject, deadline, priority,
           UUID.randomUUID().toString(), State.NOT_YET_STARTED);
   }

   public ToDo(ToDo todo) {
      this(todo.subject, todo.deadline, todo.priority, todo.id, todo.state);
   }

   public String getSubject() { return subject; }
   public String getDeadline() { return deadline; }
   public Priority getPriority() { return priority; }
   public State getState() { return state; }
   public void setState(State state) { this.state = state; }
   public int hashCode() { return id.hashCode(); }
```

```
    public boolean equals(Object o) {
        return o instanceof ToDo && [ a ];
    }

    public String toString() {
        return String.format("主題: %s, 期限: %s, 優先度: %s, 状態: %s",
                             subject, deadline, priority, state);
    }
}
```

〔プログラム2〕

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

public class ToDoList {
    private List<ToDo> todoList = new ArrayList<ToDo>();

    public void add(ToDo todo) {
        if ( [ b ] ) {
            todoList.add(new ToDo(todo));
        }
    }

    public void update(ToDo todo) {
        int index = todoList.indexOf(todo);
        if (index [ c ] ) {
            todoList.set(index, todo);
        }
    }

    public List<ToDo> select(Condition... conditions) {
        List<ToDo> result = new ArrayList<ToDo>();
        for (ToDo todo : todoList) {
            [ d ];
            for (Condition condition : conditions) {
                selected [ e ] condition.test(todo);
                if (!selected) break;
            }
            if (selected) {
                result.add(new ToDo(todo));
            }
        }
```

```
        return result;
    }
}
```

〔プログラム3〕

```
public interface Condition {
    boolean test(ToDo todo);
}
```

〔プログラム4〕

```
public class ToDoListTester {
    public static void main(String[] args) {
        ToDoList list = new ToDoList();
        list.add(new ToDo("メール送信", "201604181500", ToDo.Priority.HIGH));
        list.add(new ToDo("ホテル予約", "20160420", ToDo.Priority.LOW));
        list.add(new ToDo("チケット購入", "20160430", ToDo.Priority.MIDDLE));
        list.add(new ToDo("報告書作成", "20160428", ToDo.Priority.HIGH));
        list.add(new ToDo("会議室予約", "201605301200", ToDo.Priority.HIGH));
        list.update(new ToDo("PC購入", "20160531", ToDo.Priority.HIGH));
        for (ToDo todo : list.select()) {
            todo.setState(ToDo.State.STARTED);
            list.update(todo);
        }
        Condition condition1 = new Condition() {
            public boolean test(ToDo todo) {
                return todo.getDeadline().compareTo("20160501") < 0;
            }
        };
        Condition condition2 = new Condition() {
            public boolean test(ToDo todo) {
                return todo.getPriority().equals(ToDo.Priority.HIGH);
            }
        };
        for (ToDo todo : list.select(condition1, condition2)) {
            System.out.println(todo);
        }
    }
}
```

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　((ToDo) o).id.equals(id)　　イ　(ToDo) o.id.equals(id)

ウ　id.equals(id)　　エ　o.id.equals(id)

bに関する解答群

ア　!todoList.contains(todo)　　イ　!todoList.isEmpty()

ウ　todoList.contains(todo)　　エ　todoList.isEmpty()

cに関する解答群

ア　!= -1　　イ　< todoList.size()

ウ　== -1　　エ　>= todoList.size()

dに関する解答群

ア　boolean selected = false　　イ　boolean selected = true

ウ　int selected = 0　　エ　int selected = todoList.size()

eに関する解答群

ア　+=　　イ　=　　ウ　==　　エ　|=

設問2　プログラム 4 の実行結果を図 1 に示す。［　　］に入れる正しい答えを，解答群の中から選べ。

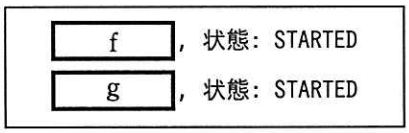

図 1　プログラム 4 の実行結果

解答群

ア　主題: PC 購入, 期限: 20160531, 優先度: HIGH

イ　主題: 会議室予約, 期限: 201605301200, 優先度: HIGH

ウ　主題: チケット購入, 期限: 20160430, 優先度: MIDDLE

エ　主題: 報告書作成, 期限: 20160428, 優先度: HIGH

オ　主題: ホテル予約, 期限: 20160420, 優先度: LOW

カ　主題: メール送信, 期限: 201604181500, 優先度: HIGH