錬成問題

▪ 以下に示すのは、キーボードから読み込んだ値が1であるか2であるかによって、異なる例外を送出するとともに、その例外を捕捉して対処を行うプログラムである。

```
import java.util.Scanner;

//---- 自作の例外[1] ---//
class Exception1 [ (1) ] Exception {
   Exception1(String s, [ (2) ] e) { super(s, e); }
}

//---- 自作の例外[2] ---//
class Exception2 [ (1) ] RuntimeException {
   Exception2(String s, [ (2) ] e) { super(s, e); }
}

public class Tester {

   //--- swの値に応じて異なる例外を送出 ---// 1
   static void work(int sw) [ (3) ] Exception {
      switch (sw) {
       case 1: [ (4) ] new RuntimeException("例外[1]発生!!");
       case 2: [ (4) ] new Exception("例外[2]発生!!");
      }
   }

   //--- workを呼び出す ---//                    2
   static void test(int sw) [ (3) ] Exception {
      [ (5) ] {
         work(sw);
      }                                                    3
      [ (6) ] (RuntimeException e) {
         [ (7) ] new Exception1("例外[1]", e);
      } [ (8) ] (Exception e) {                            4
         [ (9) ] new Exception2("例外[2]", e);
      }
   }

   public static void main(String[] args) {
      Scanner stdIn = new Scanner(System.in);

      System.out.print("sw : ");
      int sw = stdIn.nextInt();

      [ (10) ] {
         test(sw);
      } [ (11) ] ( [ (12) ] e) {
         // ここではExcpetion1とException2の両方を捕捉
         System.out.println("例外          : " + e);
         System.out.println("例外の原因 : " + e.[ (13) ]());   5
         e.[ (14) ](); // 例外の伝播の様子を表示
      } [ (15) ] {
         System.out.println("プログラム終了");                 6
      }
   }
}
```

▪ このプログラム中、[　　]で囲まれた**1**と**2**は、[　(16)　]と呼ばれる。なお、これらのうち、省略できるものは[　(17)　]。

▶ [　(17)　]の選択肢：(a) 前者　(b) 後者　(c) 両者　(d) ない

▪ **Exception** は[　(18)　]例外と呼ばれ、**RuntimeException** は[　(19)　]例外と呼ばれる。これらの二つのクラスは、いずれも[　(20)　]パッケージに所属する。なお、これらの二つのクラスのうち、上位クラスは、[　(21)　]である。

▶ [　(21)　]の選択肢：(a) 前者　(b) 後者

▪ すべての例外クラスの最上位に位置するのは、[　(22)　]クラスである。
例外は、[　(23)　]と原因という、少なくとも二つの情報を有する[　(22)　]クラスのインスタンスである。

▪ **3**は[　(24)　]ブロックと呼ばれ、**4**は[　(25)　]節あるいは例外[　(26)　]と呼ばれる。

▪ **5**の実行によって表示される実行結果を示せ。

※キーボードから読み込んだ値が

1 のときの実行結果

(27)

2 のときの実行結果

(28)

▪ [　(14)　]メソッドの呼出しによって、例外の伝播の様子が表示される。例外がメソッドをまたがって伝播している様を[　(29)　]という。

▪ **6**が実行されるのは、キーボードから読み込んだ値が[　(30)　]のときである。

▶ [　(30)　]の選択肢：(a) 1　(b) 2　(c) 1 のときと 2

▪ クラス *Exception1* に対する対処を行っているかどうかは、コンパイル時に[　(31)　]。
クラス *Exception2* に対する対処を行っているかどうかは、コンパイル時に[　(32)　]。

▶ 共通の選択肢：(a) 検査される　(b) 検査されない

▪ 空参照を通じた、フィールドのアクセスやメソッド呼出しなどを行おうとしたときや、空参照を例外として送出しようとしたときなどに送出されるのは、[　(33)　]クラス型の例外である。この例外に対する対処は、[　(34)　]。

▶ [　(34)　]の選択肢：(a) 必須である　(b) 必須ではない

▪ 配列に対して、不正なインデックスを適用したときに送出されるのは、[　(35)　]クラス型の例外である。この例外に対する対処は、[　(36)　]。

▶ [　(36)　]の選択肢：(a) 必須である　(b) 必須ではない

▪ プログラムが期待するものとは異なる事態や、通常の範囲では想定していない、あるいは想定できないような事態は、 (37) と呼ばれる。

▪ (37) を発生することを“ (38) する”といい、 (38) された (37) を検出することを“ (39) する”という。

▪ 対処が必須の (37) は、 (40) (37) と呼ばれ、 (41) クラスの直接あるいは間接の下位クラスとして実現される。

▪ 対処が必須ではない (37) は、 (42) (37) と呼ばれ、 (43) クラスの直接あるいは間接の下位クラスとして実現される。

▪ **Throwable** クラスの直接下位クラス（サブクラス）は、 (44) クラスと (45) クラスである。なお、前者のクラスの (37) が発生した場合、プログラムとしては回復を期待できない。

▪ 以下に示すのは、sw が 1 であれば **Exception** 例外を送出するメソッドである。

```
void func(int sw) throws (46) {
   if (sw == 1)throw (47) Exception();
}
```