プログラミングA 第7回・宿題　回答

1　宿題 1

参考資料：A.java

```
class A {
   static int counter = 0;
   A() {
      counter++;
   }
   String m() {
      return "a";
   }
   static int getCounter() {
      return counter;
   }
}
```

参考資料：B.java

```
class B extends A {
   A child;
   B(A a) {
      child = a;
   }
   String m() {
      return "b" + child.m() + "b";
   }
}
```

2　宿題 2-1

- number メソッドに文字列”10”が与えられると、number メソッド内で整数に変換できるため、例外は投げられず finally 節を実行して main メソッドに戻り、整数としての 10 を標準出力に出力して終了する。
- number メソッドに文字列”XXXXXXXXX” が与えられると、number メソッド内で整数に変換できないため、NumberFormatException が投げられて catch 節が実行され-1 を return するが、呼び出し元の main メソッドの処理を続行する前に finally 節を実行したうえで main メソッドにおいて整数としての-1 を標準出力に出力して終了する。finally 節は、catch 節が実行されて return しても、必ず最後に実行される。

3　宿題 2-2

参考資料：ExceptionHomework.java

```
public class ExceptionHomework {
   static int number(String s) throws OriginalException {
      try {
         System.out.println("try");
         return Integer.parseInt(s);
      } catch(NumberFormatException nfe) {
         System.out.println("catch");
         throw new OriginalException();
         //return -1;
      } finally {
         System.out.println("finally");
      }
   }
   public static void main(String[] args) {
      try {
         System.out.println(number("10"));
         System.out.println(number("XXXXXXXXX"));
      } catch(OriginalException oe) {
         System.out.println(oe);
      }
   }
}
```

参考資料：OriginalException.java

```
class OriginalException extends Exception {}
```

4　宿題 3

(3-1) P1, P2, P3。gc の時点でローカル変数 p3 から P3 が参照され続けており、P3 が P1 を、P1 が P2 をそれぞれフィールド friend により参照しているため、P1 P3 のすべてを辿ることができる。

(3-2) 該当なし。P1 P3 は循環参照していたが (P1 が P2 を, P2 が P3 を, P3 が P1 をそれぞれ参照)、それらのいずれも main メソッド内や GcHomework クラスから参照されていなかったため、プログラムから利用することができず全てガーベッジコレクタにより収集される。

5　宿題 4

(4-1)
- プログラムの誤りを通知してそれに対処することを目的とする。
- チェックされない例外は配列アクセスや数値への変換などの様々な箇所で発生する可能性があり、

必要に応じてキャッチして対処する処理を記述しておく。

- チェックされる例外は、入出力のような実行時に誤りを生じやすい処理を実施している箇所で発生する可能性があり、必ず明示的にキャッチして対処する処理を記述しておく。

(4-2) チェックされない例外とチェックされる例外がある。

- 前者は、Error クラス、RuntimeException クラスおよびそれらのサブクラスである。
- 後者は、Exception クラスおよび (RuntimeException を除く)Exception のサブクラスである。

(4-3)

- どこからも参照されていないためプログラムから利用できないヒープ上のインスタンスや配列を除去してメモリを確保することを目的とする。
- ヒープの空が不足してきたときに JavaVM により自動的に実施される。
- ただし自動的な実施のタイミングや予想困難なため、プログラム中で開発者が好ましいと考えるタイミングで強制的に実施することもできる。