■問題 1(計 90 点・部分点有り)

ロールプレイングゲーム（RPG）のソフトウェアを考える。定義するクラス，インタフェイスは以下の通りである。設問に答えよ。

(a)RPG に登場するキャラクター一般を表す public な抽象クラス RPGCharacter。ファイル RPGCharacter.java で定義され，パッケージ com.tuis.rpg.character に所属する。このクラスは次のメンバを持つ。

- 生成された RPGCharacter 型オブジェクトの総数を表す int 型の private な static フィールド num(初期値は 0)
- 体力を表す private な int 型フィールド hp
- 魔法力を表す private な int 型フィールド mp
- num の public で static なゲッタ int getNum( )
- hp の public なゲッタ int getHp( )，hp の public なセッタ void setHp(int hp)
- mp の public なゲッタ int getMp( )，mp の public なセッタ void setMp(int mp)
- 仮引数 c に渡された他の RPGCharacter 型オブジェクトに攻撃をする public な抽象メソッド void attack(RPGCharacter c)
- 仮引数 damage に渡されたダメージ量を軽減して，軽減しきれなかったダメージの分だけ自分の hp を減らすという防御行為を表した public な抽象メソッド void defence(int damage)
- フィールド hp，mp を初期化し，num の値を 1 増やす public なコンストラクタ RPGCharacter(int hp, int mp)

(b)「描画が可能である」という性質を表す public なインタフェイス Drawable。ファイル Drawable.java で定義され，パッケージ com.tuis.rpg.system に所属する。このインタフェイス Drawable は次のメンバを持つ。

- 「そのオブジェクトを描画する」という処理を表す抽象メソッド void draw( )

(c)「騎士」を表す public なクラス Knight（RPGCharacter のサブクラスで，Drawable を実装する)。ファイル Knight.java で定義され，パッケージ com.tuis.rpg.character に所属する。このクラスは次のメンバを持つ。

- RPGCharacter のコンストラクタを呼び出して，フィールド hp，mp を初期化する public なコンストラクタ Knight(int hp, int mp)
- 自分の hp の 1/5 だけ攻撃相手（仮引数 c ）の defence メソッドに渡すようにオーバライドした public な attack メソッド。
- 受けたダメージ量（仮引数 damage ）から自分の hp の 1/5 だけ引いた値が 0 以上の場合には，その値の分だけ自分の hp を減らすようにオーバライドした public な defence メソッド。
- 騎士を表す文字列 "騎士" を System.out.println メソッドで表示するようにオーバライドした public な draw メソッド

(d)「魔法使い」を表す public なクラス Mage（RPGCharacter のサブクラスで，Drawable を実装する)。ファイル Mage.java で定義され，パッケージ com.tuis.rpg.character に所属する。このクラスは次のメンバを持つ。
次の様なメンバを持つ。

- RPGCharacter のコンストラクタを呼び出して，フィールド hp，mp を初期化する public なコンストラクタ Mage(int hp, int mp)
- 自分の mp の 1/5 だけ攻撃相手（仮引数 c ）の defence メソッドに渡すようにオーバライドした public な attack メソッド。
- 受けたダメージ量（仮引数 damage ）から自分の mp の 1/5 だけ引いた値が 0 以上の場合には，その値の分だけ自分の hp を減らし，さらに自分の mp を 1 だけ減らすようにオーバライドした public な defence メソッド。
- 魔法使いを表す文字列 "魔法使い" を System.out.println メソッドで表示するようにオーバライドした public な draw メソッド

(e) 動作テストを行う public なクラス Test。このクラスはファイル Test.java で定義され，その内容は解答用紙 2 枚目の冒頭のようになっている。

設問（1）以上のクラス及びインタフェイス（a)〜(d) が書かれた各ファイルの内容を，解答欄（1)-(a)〜(d) に書け。必要に応じて，package 宣言や import 宣言を記述するのを忘れないように。また，あるクラスの private メンバの内容を他のクラスから利用する場合には，用意されたアクセッサを使うことも忘れないこと。なお，(e) の Test クラスが定義されている Test.java の内容は解答用紙 2 枚目に記載されているので参考にせよ。

設問（2）RPGCharacter 型オブジェクトのステータス（hp，mp）を設定するファイル setting.txt を読み込み、その内容に応じたオブジェクトを生成して戦闘できるよう main メソッドを修正せよ。解答欄（2）に記述せよ。なお、設定ファイルのフォーマットは以下のとおりとする。

「Type(K or M), hp, mp」

例）

```
K,50,20
M,40,60
```

※キャラクター毎に型（Knight or Mage)，体力，魔法力をカンマ区切りで 1 行 1 キャラクターで定義する。

※対応できるキャラクターの数は 2 でよい。ただし、騎士 vs 騎士、魔法使い vs 魔法使いなどにも対応できること。

※カンマで区切られた文字列（例："A,BCD,EF"）を分解するには，String クラスの 非 static メソッド split() を使うと良い（下記参照)。

```
public String[ ] split(String regex)
```

※その String 型オブジェクトが表す文字列を，第 1 引数の文字列で区切り，複数の文字列に分解した結果を String 型配列として返す。

例： String s = "Wakabaku";

```
String [ ] r = s.split( "a" );  // r[0] は "W", r[1] は "k", r[2] は "b", r[3] は"ku" となる。
```

※文字列の比較には String 型の非 static メソッド equals( ) が使用できる。これはそのオブジェクトが表す文字列と引数の文字列が等しいと true を返す。

※数字からなる文字列を int 型数値に変換するには，Integer.parseInt( ) メソッドが使える。Integer.parseInt("123") は int 型数値 123 を返す。

■問題 2(計 10 点・部分点有り)

以下は，あるクラスのメソッド foo( ) の定義を抜き出したものである。このメソッドは MyException 型例外を投げる。MyException は java.lang.Eception のサブクラスとする。空欄に当てはまる内容を書け。

```
public void foo( ) [      ] {  throw new MyException( );  }
```

■問題 1　　※スペースが足りない場合は，「裏へ続く」と書いて裏面に続きを書け。

※配点されている各パートは，些細な間違い(スペルミス，閉じカッコやセミコロン等）は各-1点の減点。ただし下線部分点がついてる部分は間違えていたらその分を減点。但しパート点は原点累積してもマイナスには成らない。

解答欄 (1)-(a) RPGCharacter.java の内容 25 点 (部分点有り)

```
package com.tuis.rpg.character;                                  1点

部分点3
public abstract class RPGCharacter {                            5点

  private static int num = 0;                                    各1点(3)
  private int hp;
  private int mp;

  public static int getNum( ) {                                  各2点(16)
    return num;
  }

  public int getHp( ) {
    return hp;
  }
  public void setHp( int hp ) {
    this.hp = hp;
  }

  public int getMp( ) {
    return mp;
  }
  public void setMp( int mp ) {
    this.mp = mp;
  }

  public abstract void attack( RPGCharacter c );

  public abstract void defence( int damage );

  public RPGCharacter( int hp, int mp ) {
    this.hp = hp; this.mp = mp; num++;
  }

}
```

解答欄 (1)-(b) Drawable.java の内容 5 点 (部分点有り)

```
package com.tuis.rpg.system;                                     1点

部分点2
public interface Drawable {                                      3点
  void draw( );                                                  1点
}
```

解答欄 (1)-(c) Knight.java の内容 15 点 (部分点有り)

```
package com.tuis.rpg.character;                                  1点
import  com.tuis.rpg.system.Drawable;                            1点

部分点2
public class Knight extends RPGCharacter                        5点
                    implements Drawable {

  部分点2
  public Knight( int hp, int mp ) {                              各2点(8)
    super( hp, mp );
  }

  public void attack( RPGCharacter c ) {
    c.defence( getHp() / 5 );
  }

  public void defence( int damage ) {                            ※アクセッサをちゃんと使用しているかに注意(左下線部)
    int d = damage - getHp()/5;
    if( d >= 0 ) {
      setHp( getHp() - d );
    }
  }

  public void draw() {
    System.out.println( "騎士" );
  }
}
```

解答欄 (1)-(d) Mage.java の内容 15 点 (部分点有り)

```
package com.tuis.rpg.character;                                  1点
import  com.tuis.rpg.system.Drawable;                            1点

部分点2
public class Mage extends RPGCharacter                           5点
                  implements Drawable {

  部分点2
  public Mage( int hp, int mp ) {                                各2点(8)
    super( hp, mp );
  }

  public void attack( RPGCharacter c ) {
    c.defence( getMp() / 5 );
  }

  public void defence( int damage ) {                            ※アクセッサをちゃんと使用しているかに注意(左下線部)
    int d = damage - getMp()/5;
    if( d >= 0 ) {
      setHp( getHp() - d );
      setMp( getMp() - 1 );
    }
  }

  public void draw() {
    System.out.println( "魔法使い" );
  }
}
```

参考：Test.java の内容 (e)

```
import com.tuis.rpg.character.RPGCharacter;
import com.tuis.rpg.system.Drawable;

public class Test {

  static void letItFight( RPGCharacter c1,
                          RPGCharacter c2 ) {
    while( c1.getHp() > 0 && c2.getHp() > 0 ) {
      c1.attack( c2 ); c2.attack( c1 );
    }
  }

  static void letItDraw( Drawable d ) {
    d.draw();
  }

  public static void main( String args[] ) {
    Knight k = new Knight( 50, 10 );
    Mage   m = new Mage( 30, 40 );
    letItDraw( k ); letItDraw( m );
    System.out.println( "現在のキャラクターの総数は"
       + RPGCharacter.getNum() );
    letItFight( k, m );
  }

}
```

■問題 1　解答欄 (2) 30 点 (部分点有り)

```
public static void main( String args[ ] ) throws IOException {
  FileInputStream fis = null; InputStreamReader isr = null; BufferedReader br = null;

  try {
      fis = new FileInputStream( "setting.txt" );            // 元のファイルストリームをオープン
      isr = new InputStreamReader( fis, "JISAutoDetect" );   // InputStreamReaderでラップする
      br  = new BufferedReader( isr );                       // さらにBufferedReaderでラップする
      String [ ] lines = new String[ 2 ];
      lines[ 0 ] = br.readLine();
      lines[ 1 ] = br.readLine();
      RPGCharacter [ ] characters = new RPGCharacter[ 2 ];
      for( int i = 0; i < 2; i++ ) {
          String [ ] tokens = lines[ i ].split( "," );
          int hp = Integer.parseInt( tokens[ 1 ] );
          int mp = Integer.parseInt( tokens[ 2 ] );
          if( tokens[ 0 ].equals( "K" ) ) {
              characters[ i ] = new Knight( hp, mp );
          }
          if( tokens[ 0 ].equals( "M" ) ) {
              characters[ i ] = new Mage( hp, mp );
          }
      }
      letItFight( characters[ 0 ], characters[ 1 ] );
    }
    catch( UnsupportedEncodingException uee ) { System.out.println( uee ); }
    catch( IOException ioe ) {  System.out.println( ioe ); }
    finally { // finallyブロックでクローズ。この場合，クローズ時の例外はメソッド外へ伝搬
      // ラッパストリームを先にクローズすること！
      if( br  != null ) br.close();  // 処理が成功したときも例外発生時もクローズする
      if( isr != null ) isr.close(); // 処理が成功したときも例外発生時もクローズする
      if( fis != null ) fis.close(); // 処理が成功したときも例外発生時もクローズする
    }
}
```

■採点基準
以下の手順のプログラムが書けていれば、各手順5点（計30点）とする。解き方は何通りかあるはずなので、以下の項目と異なっていても正しい結果が導ければ正解とする。その他，間違い一箇所につき-1減点。

1. ファイル　characters_setting.txt を読み込み、1行毎に配列に格納する。
2. 1の配列をカンマ (,) 区切りで Split して、配列に格納する。
3. 2の配列をもとに、RPGCharacter のオブジェクトを生成する。
※生成するオブジェクトは2つでもOK
4. 3で生成したオブジェクトを利用して、letItFight() メソッドを実行する。
5. ファイルのオープンと入力処理をtryブロック内で行っている。
6. 適切なcatchブロックとfinallyブロック内のクローズ処理が記述されている。

■問題 2　10 点 (部分点有り)

```
throws MyException
```