JAVA：作业2

刘泓尊 2018011446 计84 liu-hz18@mails.tsinghua.edu.cn

第一题：基础知识

按照题目要求编写代码即可。`javap -p Man`的输出结果为：

```
Compiled from "Man.java"
class Man extends BasePerson implements Person {
  private final java.lang.String name;
  private final java.lang.String description;
  protected int count;
  Man(java.lang.String, java.lang.String);
  protected void move();
  public java.lang.String getName();
  public java.lang.String getDescription();
  public int changeSomething();
}
```

`javap -p SuperMan`的输出结果为

```
Compiled from "Man.java"
class SuperMan extends Man {
  SuperMan(java.lang.String, java.lang.String);
  SuperMan();
  void fly();
  protected void move();
  public int changeSomething();
}
```

1.3

对于`man`可调用的方法，测试代码为

```
Man man = new Man("man","nothing");
System.out.println(man.getName());
System.out.println(man.getDescription());
System.out.println(man.changeSomething());
man.move(); //method1
```

运行结果为：

```
man
nothing
-1
I'm moving...
```

对于`superman`可调用的方法，测试代码为

```
SuperMan superman = new SuperMan("superman","nothing");
System.out.println(superman.getName());  //Man
System.out.println(superman.getDescription()); //method2
System.out.println(superman.changeSomething()); //SuperMan
superman.move();
superman.fly(); //method3, Superman
```

运行结果为：

```
superman
nothing
1
I'm flying...
Fly! SuperMan!
```

对于`pman`能调用的方法，测试代码为：

```
Person pman = new Man("pman","nothing");
System.out.println(pman.getName()); //Man
System.out.println(pman.getDescription());
System.out.println(pman.changeSomething());
// pman.move(); //error
```

运行结果为

```
pman
nothing
-1
```

对于`psman`能调用的方法，测试代码为

```
Person psman = new SuperMan("psman","nothing");
System.out.println(psman.getName());  //Man
System.out.println(psman.getDescription());
System.out.println(psman.changeSomething()); //method4, Person
// psman.move(); //error
// psman.fly();  //error
```

运行结果为

```
psman
nothing
1
```

对于`msman`可调用的方法，测试代码为：

```
Man msman = new SuperMan("msman","nothing");
System.out.println(msman.getName()); //Man
System.out.println(msman.getDescription());
System.out.println(msman.changeSomething()); //method5
msman.move(); //Man, SuperMan
// msMan.fly(); //error
```

运行结果为

```
msman
nothing
1
I'm flying...
```

所以，对象实例调用的方法，声明要在此方法中出现，而运行时执行的方法定义则是由引用在内存中的实际类型决定的。

1.4

（1）错误，父类实例不可转换为子类。

（2）可行

（3）错误，父类实例不可转换为子类。

第二题：单例模式

单例模式要求：

1. 单例类只能有一个实例，因此要声明为`static`
2. 单例类必须给所有其他对象创建自己的唯一实例，因此构造函数应为私有，防止外部类实例化
3. 单例类必须给所有其他对象提供这一实例，因此要设计公有静态方法，返回这一实例。

此外，可以设计为`Instance`被装载时便实例化。这样可以保证线程安全，`java`的`classloader`机制避免了多线程的同步问题。但是不能实现懒惰加载。

```
class Singleton extends BaseSingleton {
    private static Singleton mInstance = new Singleton();
    private Singleton() {}
    public static Singleton getInstance() {
        return mInstance;
    }
}
```

第三题：考勤记录

设计了`Student`类和`Teacher`类 派生`BaseStaff`基类。

其中`Student`类的方法和属性如下：

```
class Student extends BaseStaff {
  private final int mNumber;
  private final java.lang.String mName;
  private final java.lang.String mSex;
  private final int mAge;
  private final java.lang.String mSignYear;

  public Student(int, java.lang.String, java.lang.String, int,
java.lang.String);
  public Student(int);
  public final java.lang.String getType();
  public final int getNumber();
  public final java.lang.String toString();
}
```

`Tearcher`类的方法和属性如下：

```
class Teacher extends BaseStaff {
  private final int mNumber;
  private final java.lang.String mSex;
  private final int mAge;
  private final java.lang.String mMajor;

  public Teacher(int, java.lang.String, int, java.lang.String);
  public Teacher(int);
  public final java.lang.String getType();
  public final int getNumber();
  public final java.lang.String toString();
}
```

对于学生/老师信息以及打卡过程，我抽象为`login`和`punch`两个过程，由类`StaffManager`负责管理。

```
class StaffManager {
  private static java.util.HashMap<BaseStaff, NaiveStaff> hmap;
  StaffManager();
  public static void login(java.lang.String);
  public static void punch(java.lang.String);
  public static final BaseStaff maxPunchStuff();
  static {};
}
```

`StaffManager`类维护了一个`HashMap<BaseStaff, NaiveStaff>`来保存成员信息和打卡次数。每有一次login或punch，该类便会创建一个`BaseStaff`实例作为键，若此键已存在，则说明是打卡操作，则将打卡次数+1。

```
    public static void punch(final String info) {
        String[] splitInfos = info.split(" ");
        if (splitInfos[0].equals("T")) {
            hmap.get(new Teacher(Integer.parseInt(splitInfos[1]))).punch();
        } else {
            hmap.get(new Student(Integer.parseInt(splitInfos[1]))).punch();
        }
    }
```

最终`StaffManager.maxPunchStuff`方法负责获得打卡次数最多的`BaseStaff`对象，由派生类的`toString()`方法保证可以输出结果。