Java05

Task1

1.

Person类代码如下：

```
class Person {
    private String name;
    private int age;
    private int sex;

    public Person(Person another){//复制对象
        this.name = another.name;
        this.age = another.age;
        this.sex = another.sex;
    }
    public Person(String name,int age,int sex){
        this.name = name;
        this.age = age;
        this.sex = sex;
    }
    private void eat() {
        System.out.println(name+"正在吃东西");
    }

    private void sleep() {

    }

    private void dadoudou() {

    }
}
```

this关键字的作用

主要有三个作用： **一、** 当方法的形参或局部变量与类的成员变量同名时，为了明确指定要使用的是成员变量而不是局部变量，就必须使用this。 **二、** 在一个构造方法中，可以使用this()来调用同一个类中的另一个构造方法。 *（注：this()必须是构造方法中的第一条语句，且只能在构造方法中使用this()，不能在普通方法中使用。）* **三、** 当需要在一个方法中将当前对象本身传递给另一个方法时，可以使用this。

2.

代码如下：

```java
import java.util.Scanner;
public class java05 {
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        String name = sc.nextLine();
        int age = sc.nextInt() , sex = sc.nextInt();
        Person person1 = new Person(name ,age ,sex); //字段赋值
        Person person2 = new Person(person1);
    }
}
class Person {
    private String name;
    private int age;
    private int sex;

    public Person(Person other){
        this.name = other.name;
        this.age = other.age;
        this.sex = other.sex;
    }
    public Person(String name,int age,int sex){
        this.name = name;
        this.age = age;
        this.sex = sex;
    }
    private void eat() {
        System.out.println(name+"正在吃东西");
    }

    private void sleep() {

    }

    private void dadoudou() {

    }
}
```

类和对象的关系

- 类是模板，对象是根据模板创建出来的具体实例。
- 类是编译时的代码结构，程序运行时不分配内存，对象则是运行时的内存实体 *（即类在编译时存在，对象在运行时存在）* 。
- 类包含字段声明和方法定义，而对象中的字段有具体值，且可通过对象调用方法。
- 一个类可以创建多个对象，每个对象都是独立的。

3.

为Person类的字段和方法添加我认为合适的访问修饰符

```
private String name
private int age
private int sex
public void eat()
public void sleep()
public void dadoudou()
```

各种访问修饰符的限制范围

访问修饰符	限制范围
public	同类+同包+子类+不同包
protected	同类+同包+同包子类
默认（无修饰符）	同类+同包
private	同类

Task2

一个小总结：

成员变量（实例变量） vs 静态变量（类变量）

特性	成员变量 (实例变量)	静态变量 (类变量)
关键字	无特定关键字	`static`
归属	属于各个对象实例	属于类本身
内存分配	每个对象创建时（`new`）， 都会在堆内存中拥有自己独立的一份	在类加载时即在方法区产生，全类共有一份
生命周期	随着对象的创建而创建， 随着对象被垃圾回收而销毁	随着类的加载而产生，随着程序结束而销毁
调用方式	对象名.变量名	类名.变量名
数据特性	各个对象的数据独立，修改互不影响	所有对象共享同一数据，一处修改， 所有对象看到的都改变
用途	描述对象独有的状态	描述所有对象共有的状态或保存类级别的数据

实例方法 vs 静态方法

特性	实例方法	静态方法
关键字	无特定关键字	`static`
归属	属于对象实例	属于类本身
调用方式	必须通过对象调用	通过类名直接调用
访问权限	可以直接访问和修改实例变量和实例方法， 可以访问和修改静态变量和静态方法	不可以直接访问实例变量或调用实例方法 (因为没有`this`)， 可以访问静态变量和调用其他静态方法
用途	定义对象的行为或功能， 这些操作通常依赖于对象的具体状态	实现工具类方法、工厂方法或辅助函数， 这些方法与对象状态无关

- 实例方法可以通过`this`关键字访问当前对象实例，而静态方法没有`this`。