package com.javarush.test.level11.lesson06.task01;

/\* Лошадь и пегас  
Написать два класса: Horse (лошадь) и Pegas (пегас).  
Унаследовать пегаса от лошади.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Horse  
 {  
  
 }  
  
 public class Pegas   
 {  
  
 }  
}

package com.javarush.test.level11.lesson06.task02;  
  
/\* Домашние животные  
Написать три класса: Pet (домашнее животное), Cat(кот) и Dog(собака).  
Унаследовать кота и собаку от животного.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Pet  
 {  
  
 }  
  
 public class Cat  
 {  
  
 }  
  
 public class Dog  
 {  
  
 }  
}

package com.javarush.test.level11.lesson06.task03;  
  
/\* В гостях у бабушки  
Написать шесть классов: Animal (животное), Cow(корова) и Pig(свинья), Sheep(овца), Bull(бык), Goat(козел).  
Унаследовать корову, свинью, овцу, быка и козла от животного.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Animal  
 {  
  
 }  
  
 public class Cow  
 {  
  
 }  
  
 public class Pig  
 {  
  
 }  
  
 public class Sheep  
 {  
  
 }  
  
 public class Bull  
 {  
  
 }  
  
 public class Goat  
 {  
  
 }  
  
}

package com.javarush.test.level11.lesson06.task04;  
  
/\* Все мы работники  
Написать четыре класса: Employee(сотрудник), Manager(управляющий), Chief(директор) и Secretary(секретарь).  
Унаследовать управляющего, директора и секретаря от сотрудника.  
\*/  
  
public class Solution  
{  
 public class Manager  
 {  
  
 }  
  
 public class Chief  
 {  
  
 }  
  
 public class Employee  
 {  
  
 }  
  
 public class Secretary  
 {  
  
 }  
}

package com.javarush.test.level11.lesson06.task05;  
  
/\* ИТ-компания  
Написать девять классов: Worker(сотрудник), Clerk (клерк), IT (ИТ-специалист), Programmer(программист), ProjectManager(менеджер проекта), CTO(технический директор), HR(рекрутер), OfficeManager(офис-менеджер), Cleaner (уборщик).  
Унаследовать программиста, менеджера проекта и технического директора от ИТ-специалиста.  
Унаследовать рекрутера, уборщика и офис-менеджера от клерка.  
Унаследовать клерка и ИТ-специалиста от сотрудника.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Worker  
 {  
  
 }  
  
 public class Clerk  
 {  
  
 }  
  
 public class IT  
 {  
  
 }  
  
 public class Programmer  
 {  
  
 }  
  
 public class ProjectManager  
 {  
  
 }  
  
 public class CTO  
 {  
  
 }  
  
 public class OfficeManager  
 {  
  
 }  
  
 public class HR  
 {  
  
 }  
  
 public class Cleaner  
 {  
  
 }  
}

package com.javarush.test.level11.lesson08.task01;  
  
/\* Все скрыто  
Скрыть все внутренние переменные класса Cat.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Cat  
 {  
 public String name;  
 public int age;  
 public int weight;  
  
 public Cat()  
 {  
 }  
  
 public Cat(String name, int age, int weight)  
 {  
 this.name = name;  
 this.age = age;  
 this.weight = weight;  
 }  
 }  
}

package com.javarush.test.level11.lesson08.task02;  
  
/\* Доступ есть  
Скрыть все внутренние переменные класса Cat, но только те, к которым остается доступ с помощью методов.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Cat  
 {  
 public String name;  
 public int age;  
 public int weight;  
  
 public Cat(String name, int age, int weight)  
 {  
 this.name = name;  
 this.age = age;  
 this.weight = weight;  
 }  
  
 public String getName()  
 {  
 return name;  
 }  
  
 public void setName(String name)  
 {  
 this.name = name;  
 }  
  
 public int getAge()  
 {  
 return age;  
 }  
  
 public void setAge(int age)  
 {  
 this.age = age;  
 }  
 }  
}

package com.javarush.test.level11.lesson08.task03;  
  
/\* Ничего не поменяешь  
Скрыть все внутренние переменные класса Cat, а также методы, позволяющие менять внутреннее состояние объектов класса Cat.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Cat  
 {  
 public String name;  
 public int age;  
 public int weight;  
  
 public Cat(String name, int age, int weight)  
 {  
 this.name = name;  
 this.age = age;  
 this.weight = weight;  
 }  
  
 public String getName()  
 {  
 return name;  
 }  
  
 public void setName(String name)  
 {  
 this.name = name;  
 }  
  
 public int getAge()  
 {  
 return age;  
 }  
  
 public void setAge(int age)  
 {  
 this.age = age;  
 }  
 }  
}

package com.javarush.test.level11.lesson08.task04;  
  
/\* Инкапсуляция для классов Cat и Dog  
Скрыть все внутренние переменные класса Cat и Dog. Также скрыть все методы, кроме тех, с помощью которых эти классы взаимодействуют друг с другом.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 Cat cat = new Cat("Vaska",5);  
 Dog dog = new Dog("Sharik",4);  
  
 cat.isDogNear(dog);  
 dog.isCatNear(cat);  
 }  
  
 public static class Cat  
 {  
 public String name;  
 public int speed;  
  
 public Cat(String name, int speed)  
 {  
 this.name = name;  
 this.speed = speed;  
 }  
  
 public String getName()  
 {  
 return name;  
 }  
  
 public int getSpeed()  
 {  
 return speed;  
 }  
  
 public boolean isDogNear(Dog dog)  
 {  
 return this.speed > dog.getSpeed();  
 }  
 }  
  
 public static class Dog  
 {  
 public String name;  
 public int speed;  
  
 public Dog(String name, int speed)  
 {  
 this.name = name;  
 this.speed = speed;  
 }  
  
 public String getName()  
 {  
 return name;  
 }  
  
 public int getSpeed()  
 {  
 return speed;  
 }  
  
 public boolean isCatNear(Cat cat)  
 {  
 return this.speed > cat.getSpeed();  
 }  
 }  
}

package com.javarush.test.level11.lesson08.task05;  
/\* Добавь недостающие переменные  
Посмотри внимательно на методы и добавь недостающие переменные.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Cat  
 {  
  
   
  
 public Cat(String name, int age, int weight)  
 {  
   
 }  
  
 public String getName()  
 {  
return null;  
 }  
  
 public int getAge()  
 {  
 return null;  
 }  
  
 public void setWeight(int weight)  
 {  
  
 }  
  
 public void setSpeed(int speed)  
 {  
  
 }  
 }  
}

package com.javarush.test.level11.lesson11.home01;  
  
/\* Адам и Ева  
Написать два класса Adam(Адам) и Eve(Ева).  
Унаследовать Еву от Адама.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 //Адам  
 public class Adam  
 {  
  
 }  
  
 //Ева  
 public class Eve  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.home02;  
  
/\* AppleIPhone и SamsungGalaxyS2  
Написать два класса AppleIPhone и SamsungGalaxyS2.  
Унаследовать SamsungGalaxyS2 от AppleIPhone.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class AppleIPhone  
 {  
  
 }  
  
 public class SamsungGalaxyS2  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.home03;  
  
/\* Эволюция  
Написать четыре класса: Fish(Рыбы), Animal(Животные), Ape(Обезьяны), Human (Человек).  
Унаследовать животных от рыб, обезьян от животных и человека от обезьян.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Fish  
 {  
  
 }  
  
 public class Animal  
 {  
  
 }  
  
 public class Ape  
 {  
  
 }  
  
 public class Human  
 {  
  
 }  
  
}

package com.javarush.test.level11.lesson11.home04;  
  
/\* Религии  
Написать три класса: Judaism(Иудаизм), Christianity(Христианство), Islam(Мусульманство)  
Унаследовать христианство от иудаизма и мусульманство от христианства.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Judaism  
 {  
  
 }  
  
 public class Christianity  
 {  
  
 }  
  
 public class Islam  
 {  
  
 }  
  
}

package com.javarush.test.level11.lesson11.home05;  
  
/\* От школьника к рабству  
Написать четыре класса: Schoolboy(школьник), Student(студент), Worker(Сотрудник), Slave (Раб)  
Унаследовать студента от школьника, сотрудника от студента, раба от сотрудника.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Schoolboy  
 {  
  
 }  
  
 public class Student   
 {  
  
 }  
  
 public class Worker  
 {  
  
 }  
  
 public class Slave  
 {  
  
 }  
  
}

package com.javarush.test.level11.lesson11.home06;  
  
/\* Первая правильная «цепочка наследования»  
Расставь правильно «цепочку наследования» в классах: Pet (домашнее животное), Cat (кот), Dog(собака).  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Pet  
 {  
  
 }  
  
 public class Cat  
 {  
  
 }  
  
 public class Dog  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.home07;  
  
/\* Вторая правильная «цепочка наследования»  
Расставь правильно «цепочку наследования» в классах: Carnivora (плотоядное животное), Cow (корова), Dog(собака), Pig(свинья), Animal (животное).  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Carnivora  
 {  
  
 }  
  
 public class Cow  
 {  
  
 }  
  
 public class Dog  
 {  
  
 }  
  
 public class Pig  
 {  
  
 }  
  
 public class Animal  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.home08;  
  
/\* Третья правильная «цепочка наследования»  
Расставь правильно «цепочку наследования» в классах: Pet (домашнее животное), Cat (кот), Dog(собака), Car (машина).  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Pet  
 {  
  
 }  
  
 public class Cat  
 {  
  
 }  
  
 public class Car  
 {  
  
 }  
  
 public class Dog  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.home09;  
  
/\* Четвертая правильная «цепочка наследования»  
Расставь правильно «цепочку наследования» в классах: House (дом), Cat (кот), Dog(собака), Car (машина).  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class House  
 {  
  
 }  
  
 public class Cat  
 {  
  
 }  
  
 public class Car  
 {  
  
 }  
  
 public class Dog  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.home10;  
  
/\* Пятая правильная «цепочка наследования»  
Расставь правильно «цепочку наследования» в классах: House (дом), Cat (кот), Dog(собака), Car (машина), Animal(животное), Assets (имущество).  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class House  
 {  
  
 }  
  
 public class Cat  
 {  
  
 }  
  
 public class Car  
 {  
  
 }  
  
 public class Dog  
 {  
  
 }  
  
 public class Animal  
 {  
  
 }  
  
 public class Assets  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.bonus01;  
  
/\* Нужно исправить программу, чтобы компилировалась и работала  
Исправь наследование в классах: (классы Cat, Dog, Pet, House, Airplane).  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class Pet extends House  
 {  
  
 }  
  
 public class Cat extends Airplane  
 {  
  
 }  
  
 public class Dog extends Cat  
 {  
  
 }  
  
 public class House extends Dog  
 {  
  
 }  
  
 public class Airplane  
 {  
  
 }  
}

package com.javarush.test.level11.lesson11.bonus02;  
  
/\* Нужно добавить в программу новую функциональность  
Добавь общий базовый класс к классам-фигур: (фигуры из шахмат).  
\*/  
  
public class Solution  
{  
 public static void main(String[] args)  
 {  
 }  
  
 public class King  
 {  
 }  
  
 public class Queen  
 {  
 }  
  
 public class Rook  
 {  
 }  
  
 public class Knight  
 {  
 }  
  
 public class Bishop  
 {  
 }  
  
 public class Pawn  
 {  
 }  
}

package com.javarush.test.level11.lesson11.bonus03;  
  
/\* Задача по алгоритмам  
Написать метод, который возвращает минимальное и максимальное числа в массиве.  
\*/  
  
public class Solution  
{  
 public static void main(String[] args) throws Exception  
 {  
 int[] data = new int[]{1, 2, 3, 5, -2, -8, 0, 77, 5, 5};  
  
 Pair<Integer, Integer> result = *getMinimumAndMaximum*(data);  
  
 System.*out*.println("Minimum is " + result.x);  
 System.*out*.println("Maximum is " + result.y);  
 }  
  
 public static Pair<Integer, Integer> getMinimumAndMaximum(int[] array)  
 {  
 if (array == null || array.length == 0)  
 {  
 return new Pair<Integer, Integer>(null, null);  
 }  
  
 //Напишите тут ваше решение  
  
 return new Pair<Integer, Integer>(0, 0);  
 }  
  
  
 public static class Pair<X, Y>  
 {  
 public X x;  
 public Y y;  
  
 public Pair(X x, Y y)  
 {  
 this.x = x;  
 this.y = y;  
 }  
 }  
}