Mock2

This is a mock test for which you will not get a grade. Just test your knowledge.

Create the following programs. You have 70 minutes in total. **The tasks will be assessed automatically. Make sure that the names of the created classes, attributes and methods are consistent with the content of the task.**

**Don't forget to test all created classes. Create a Main class with a main method. In the main method, create objects based on the created classes, assign attribute values, call methods, display values returned by the methods.**

1. Define a **Numbers** class that contains a constructor with five int parameters that allows you to pass 5 integers to the object. Add a different() method that returns true if all numbers are different or false otherwise. Example:

sample five numbers: 3 4 2 1 6  
different() 🡪 true  
sample five numbers: 9 7 4 3 7  
different() 🡪 false

1. Define an **Arrays** class that contains a static method arr(int[] arr1, int[] arr2). The method returns true if the number of two-digit numbers contained in arr1 and arr2 are the same or false otherwise. Example:

int[] arr1 = {15,8,2,37,49,117}  
int[] arr2 = {9,6,7,12,48,4,6,90,5}  
arr(arr1,arr2) 🡪 true

1. Define a **Family** class that describes a group of people that make up a family. The constructor of the class has one parameter of the Person[] type, which allows you to pass people to the family at the time of its creation. Add an adults() method that returns the number of adults in the family (18 or older).

public class Person {  
 private String name;  
 private int age;  
 Person(String n, int a){name=n; age=a;}  
 public int getAge(){return age;}  
}

1. Define a **ShoppingList** class describing the list of products to be purchased according to the structure in the Product class. Add the add(Product product) method to add the product to the shopping list. The toString() method returns the names of the products in the shopping list, in the order they were added, separated by commas, without spaces, as a single string, and the total() method returns the total number of products to purchase. Example:

two products have been added to the shopping list: milk, 2 and apple, 4  
toString() 🡪 "milk,apple"  
total() 🡪 6

// create class Product in a separate file  
public class Product {  
 private String name;  
 private int quantity;  
 Product(String n, int q){name=n; quantity=q;}  
 public String getName(){return name;}  
 public int getQuantity(){return quantity;}  
}

1. Based on the Counter class, define a derived class **SuperCounter**. Add a constructor with an int parameter to set the initial value of the counter. Add the addN(int n) method to increment the counter value by any n value.

// create class Counter in a separate file  
public class Counter {  
 private int counter;  
 Counter(int c){counter=c;}  
 public void add1(){counter++;}  
 public int getCounter(){return counter;}  
}

1. Based on the Vehicle class, define a derived class **Car**, containing the maxSpeed attribute, of type int. Add a constructor with two parameters, in which you initialize the values of the attributes: seats and maxSpeed. Add an int[] spec() method that returns the car's specification as a two-element array. The first element of the array contains the number of seats in the car, and the second the maximum speed.  
     
   // create Vehicle class in a separate file  
   public class Vehicle {  
    private int seats;  
    Vehicle(int s){seats = s;}  
    public int getSeats(){return seats;}  
   }
2. The **Cities** class includes an attribute in the form of an array with city names. The initial value of the attribute is passed through a constructor parameter. The filter(char) method returns an object of the Cities class with those cities whose names start with the given character. The cities() method returns a string consisting of the city names contained in the object attribute. Example:

Cities(["Warszawa","Sopot","Kielce","Szczecin"]).filter('S').cities() 🡪 "SopotSzczecin"