PP3 Mock Test3

NAME, SURNAME, STUDENT ID (IN CAPITAL LETTERS): ……

Create programs in files with the names given in parentheses at the beginning of each task. If you use other names, you will not receive points.

Each function has a name f(). If you use a different name, you will not receive points.

The **Person** class contains two attributes that represent name and surname. The initial values of the attributes are passed through the constructor parameters. Specify the text representation of the object, consisting of two uppercase letters of the first name and last name. Example:

```
Person("anna","may") → "AM"
```

The **Isogram** class includes the static isogram(String) method. The method returns true if the text parameter is an isogram or false otherwise. An isogram is a string where all characters are different. Example:

```
isogram("red sun") → true
isogram("blue water") → false
isogram("BLUE water") → true
isogram("my blue water") → false
```

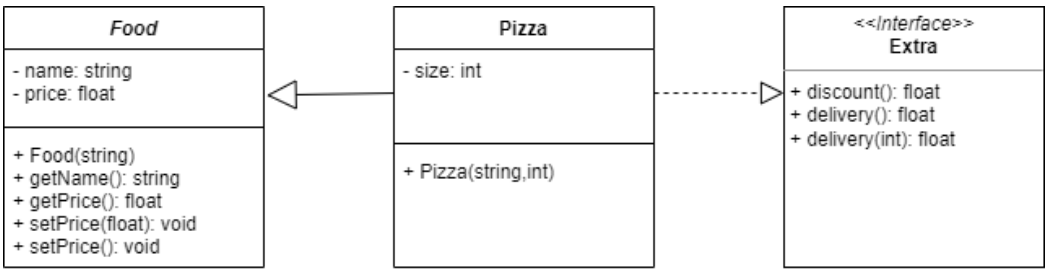
The **Logic** class contains an array attribute of logical values. The attribute is initialized through a constructor parameter. The opposite() method returns the number of array elements for which both adjacent elements have the opposite value. Example:

```
boolean arr1 = {true,false,false}
boolean arr2 = {true,false,true,false}
boolean arr3 = {true,false,true,true,false,true,false,true,false}
new Logic(arr1).opposite() => 0
new Logic(arr2).opposite() => 2
new Logic(arr3).opposite() => 5
```

The **Cities** class includes an array attribute 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:

```
String arr = {"Warszawa","Sopot","Kielce","Szczecin"}
new Cities(arr).filter('S').cities() → "SopotSzczecin"
```

Based on the UML class diagram, define the **Pizza** class. In the Food class, setPrice is an abstract method. In this method, set the price of the pizza based on its size, according to the rule: size of the pizza - 10. The discount method returns the discount for the purchased pizza, according to the rule: when the price of the pizza is at least 30, the discount is 5%. The delivery method returns the cost of pizza delivery, which is 6, unless the size of the pizza is 50 or more, then the cost of delivery is 8. The cost of delivery may also include a tip for the supplier. The tip value can be passed via a method parameter.



The **Numbers** class includes an array attribute of integers. The initial value of the attribute is passed through a constructor parameter. The class implements the interface:

```
interface Ok { boolean ok() }
```

The ok() method returns true when the even-indexed array elements contain even values and the odd-indexed array elements contain odd values. Otherwise, the method returns false. Example:

```
int[] arr1 = {6,7,6,1,4}
int[] arr2 = {2,5,2,8,4}
new Numbers(arr1).ok() → true
new Numbers(arr2).ok() → false
```

The **Product** class contains two attributes: name (String) and price (float), initialized by a constructor and all get and set methods. Implement a Comparable interface that allows you to sort Product class objects by product price, descending. Example:

```
Product[] arr =
   {new Product("milk",5), new Product("cheese",2), new Product("ham",9)}
Product[0].getPrice() → 5
Product[1].getPrice() → 2
Product[2].getPrice() → 9
Arrays.sort(arr)
Product[0].getPrice() → 9
Product[1].getPrice() → 5
Product[2].getPrice() → 2
```