Един от най-използваните елементи от ООП-то, са т.нар. **интерфейси** и **абстрактни класове**.

В езика C++ те не същестуват в същия вид, както например в езиците C#/Java,

целта на тази задача е да се създадат такива подобни.

**Comparable**

Ще играе ролята на интерфейс, чрез който ще показваме, че даден клас може да се сравнява.

* Предефинирайте операторите == и != като абстрактни функции, приемащи указател от тип Comparable

**Debug**

Ще играе ролята на интерфейс, чрез който ще даваме възможността, да принтираме допълнителна информация за

състоянието на даден клас, с цел откриване на грешки.

* Създайте абстрактна функция debug\_print, която да връща string обект.

**Serializable**

Ще играе ролята на интерфейс, показващ че даден клас може да се сериализира (да се превърне до низ), както и

десериализира (че може да се създаде обект, по подаден низ).

* Създайте абстрактна функция to\_string, която връща string обект.
* Създайте абстрактна функция from\_string, която приема string обект, и модифицира текущия обект, спрямо подадения низ.

**Пример:**

Ако имаме дадения клас Foo:

class Foo{

public:

   int value;

};

to\_string може да връща стойността на value, а from\_string ще приема един такъв низ, и ще променя

стойността на value спрямо съдържанието на низа.

Всеки един от тези класове, ще бъдат наследени от един специален клас **Object**

**Object**

Този абстрактен клас ще играе ролята на база, върху която ще стъпят всички други класове в задачата.

За целите на конкретната задача, той ще съдържа и някои член-данни.

* Наследява Comparable, Debug и Serializable
* Конструктор с параметри
* Метод clone, който прави дълбоко копие на текущия обект. Връща се Object\*
* Име на файл (name)
* Локация на файл (location)
* Разширение на файл (extension)
* Метод за създаване на пълен път на файла
  + Формат: <location>/<name>.<extension>

С помощта на Object ще създадем още два класа - **Document** и **KeyValueDatabase**

**Document**

Играе ролята на текстов документ, където можем да пишем и четем ред по ред

* Наследява Object
* Пази в себе си колекция от низове (един низ е равен на един ред)
* Метод за писане във документа write\_line
* Методи за четене в документа. Документа се чете ред по ред, като се "помни" до къде сме прочели документа
  + Метод read\_line, който връща следващия ред, който трябва да се "прочете"
  + Метод read\_line, приемащ аргумент кой ред да се прочете. След извикването на този метод, четенето на документа продължава от подадения ред
  + Ако не може да се прочете ред (например сме прочели документа до край), да се хвърля std::out\_of\_range изключение
  + Броенето на редовете започва от 1
* Два документа са равни, когато техните редове са едни и същи *(бележка: ако редовете на документ 1 е подмножество на редовете на документ 2, документ 1 НЕ Е равен на документ 2)*
* Сериализирането на един документ се извършва по следния начин:
  + На първите три реда се записва името, локацията и разширението на документа
  + На следващите реда, се записва съдържанието на документа
  + След последния ред от документа, при сериализиране се очаква да има нов ред също
* Десериализирането на един документ се извършва по същия начин
* Принтирането на допълнителната информация се извършва по следния начин:
  + За всеки ред от съдържанието, се записва нов ред Line <номер\_на\_реда>:<ред>
  + След последния ред се очаква да има нов ред​

**KeyValueDatabase**

Играе ролята на база от данни, пазеща в себе си двойки от ключ и стойност.

* Наследява Object
* Пази в себе си двойки ключ и стойност, където ключът е низ, а стойността - цяло число.

Съдържа метод за добавяне на нова двойка ключ и стойност. Ако ключът вече се съдържа в колекцията,

* се хвърля std::invalid\_argument

Съдържа метод за взимане на стойност, по подаден ключ. Ако ключът не се съдържа в колекцията,

* се хвърля std::invalid\_argument
* Две бази са равни, когато техните двойки от ключове и стойности са равни *(бележка: ако двойките ключ/стойност на база1 са подмножество на тези от база2, база 1 НЕ Е равна на база 2)*
* Сериализирането на една база се извършва по следния начин:
  + На първите три реда се записва името, локацията и разширението на документа
  + На следващите редове, се записва ред със следния формат <ключ>:<стойност>
  + Очаква се низа да завършва с нов ред
* Десериализирането на една база се случва по същия начин
* Принтирането на допълнителната информация се извършва по следния начин:
  + За всяка двойка, се записва нов ред {<ключ>:<стойност>}
  + Очаква се низа да завършва с нов ред

**За този клас е забранено използването на std::map, std::unordered\_map и всички други колекции, които предоставят търсена функционалност "наготово" (std::vector и други подобни са позволени)**