

Финален изпит No. 1

Студент:

Точки:

Инструкции:

1. Изпълнете **всички** задания.
2. Решението на **заданията** да се **предаде на floppy** и **допълнително** да се качи със **студентския акаунт на Moodle**.
2. **Използвайте дадените означения** за класове, променливи и методи.

Скала за оценяване:

- | | |
|---|--------------------|
| 2 | от 0 до 54 точки |
| 3 | от 55 до 64 точки |
| 4 | от 65 до 74 точки |
| 5 | от 75 до 84 точки |
| 6 | от 85 до 100 точки |

Решете следните задачи като спазите изискванията за **капсулиране, скриване на информация и многократно използване на код.**

Задание за програмиране

Обработката на съобщения е често срещана задача в програмирането. Създайте графично приложение на Java, което демонстрира подобни операции **в описаната по-долу последователност:**

1. **Напишете дефиницията на абстрактен клас `class Message` за моделиране на съобщение от най-прост вид.** Този клас има две променливи (*data members*)
 - Стринг променлива `source` – дефинира източник на съобщението
 - Стринг променлива `destination` – дефинира получателя на съобщението
 - Целочислена променлива `messageID` дефинира **уникален** номер за идентифициране на съобщението

Напишете:

- **SET и GET методи** за данните на класа (за `messageID` само **GET метод**)
- **Конструктор** за общо ползване и по подразбиране за `class Message` (**използвайте празен низ** по подразбиране за стойностите на данните)
- Необходимите **декларация и команди**, за да може `messageID` да приема **неповтаряща се (уникална) стойност** в конструктора на класа
- `toString()` метод за този клас за извеждане на данните на отделни редове в текстов вид с подходящи етикети

Точки:10

2. **Напишете дефиницията на `interface Appendable`.**

- Нека всяко съобщение **респ. `class Message` да е `Appendable`**, т.е. да може да се добавя към текста на друго съобщение. **Нека `class Message` онаследява `interface Appendable`**
- **Нека `interface Appendable` да съдържа единствено следния метод**
`void appendTo(Appendable obj)`

// добавя данните на текущото съобщение към съобщението представено с референцията `Appendable obj`

Точки:6

3. **Напишете дефиницията на конкретен `class EncMessage`, който онаследява `class Message` и така моделира клас от криптирани съобщения.** В допълнение към наследените променливи този клас **има** още **две** променливи (*data members*):
 - Целочислена променлива `key` която служи за криптиране на съобщението
 - Стринг променлива `txt` която представлява криптирания текст на съобщението.

Напишете следните методи в *class EncMessage* - служещи за криптиране и декриптиране на текста на съобщението

- a) *String encrypt(String text)* -> криптира текста *text* и връща получения крипт текст като се използва следния алгоритъм
1. Преобразува се стринга *text* в масив *chrArray* от символи *char* като се използва метода *toCharArray()* на *String* обекта *text* (Методът *toCharArray()* връща масив от *char* - символите на *text*)
 2. Създава се друг едномерен масив *encArray* от символи със същата дължина както броя на елементите в масива *chrArray*
 3. Елементите на *chrArray* се копират с циклично отместване в масива *encArray* като на всеки елемент с индекс *i* в масива *encArray* се присвоява елемент с индекс $(i+key) \bmod \text{len}$ от масива *chrArray*, където *len* е дължината на масива *encArray*
 4. Връща се масива *encArray*
- b) *String decrypt(String text)* -> де-криптира текста *text* и връща получения декриптирания текст като се използва циклично отместване с обратен знак по процедура аналогична на тази за криптиране

Точки:16

4. **Напишете** в *class EncMessage*:

- Необходимите **SET** и **GET** методи за данните на класа **Нека** **SET** метода за данната *txt* да приема некриптиран текст, който се криптира и после се присвоява на данната *txt* **Нека** **GET** метода за данната *txt* да връща некриптиран текст, който се съдържа в данната *txt*.
- **Трите** вида конструктори (за общо ползване, подразбиране и копиране)
- *toString()* метод за този клас, която да извежда в текстов вид всички данни на обектите на класа (текстът на **съобщението** да се извежда заградено в **квадратни скоби**)

Точки:6

5. **Напишете** дефиницията на конкретен *class TextMessage*, който **онаследява** *class Message*.: В допълнение към онаследените променливи този клас има още **една** стринг променлива *txt* която представлява **текста на съобщението** (без криптиране).

Напишете:

- Необходимият **SET** и **GET** метод за *txt*
- **Трите** вида конструктори (за общо ползване, подразбиране и копиране)
- *toString()* метод за този клас, която да извежда в текстов вид всички данни на обектите на класа (текстът на **съобщението** да се извежда заградено в **квадратни скоби**)

Точки:12

6. **Реализирайте** в *class EncMessage* метода *appendTo(Appendable obj)* на *interface Appendable*, така че да изпълнява следния алгоритъм:

Ако *obj* е *TextMessage* тогава

```
{
    //1) долепете отдясно съдържанието на текста в променливите за source,
    destination и txt на текущия обект към съдържанието на съответните
    променливи за source, destination и txt на обекта рефериран с obj като
    използвате празен символ за разделител. Текстът txt от текущия обект да се
    декриптира преди да се долепи.
}
```

Ако *obj* е *EncMessage* тогава

```
{
    //2) долепете отдясно съдържанието на текста в променливите за source,
    destination и txt на текущия обект към съдържанието на съответните
```

променливи за *source*, *destination* и *txt* на обекта рефериран с *obj* като използвате празен символ за разделител. Текстът *txt* от текущия обект и този *obj* **да се декриптират** преди да се долепят.

}

Точки:10

7. Реализирайте в *class TextMessage* метода *appendTo(Appendable obj)* на *interface Appendable*, така че да изпълнява следния алгоритъм:

Ако *obj* е *TextMessage* тогава

{

//1) долепете отдясно съдържанието на текста в променливите за *source*, *destination* и *txt* на текущия обект към съдържанието на съответните променливи за *source*, *destination* и *txt* на обекта рефериран с *obj* като използвате празен символ за разделител. .

}

Ако *obj* е *EncMessage* тогава

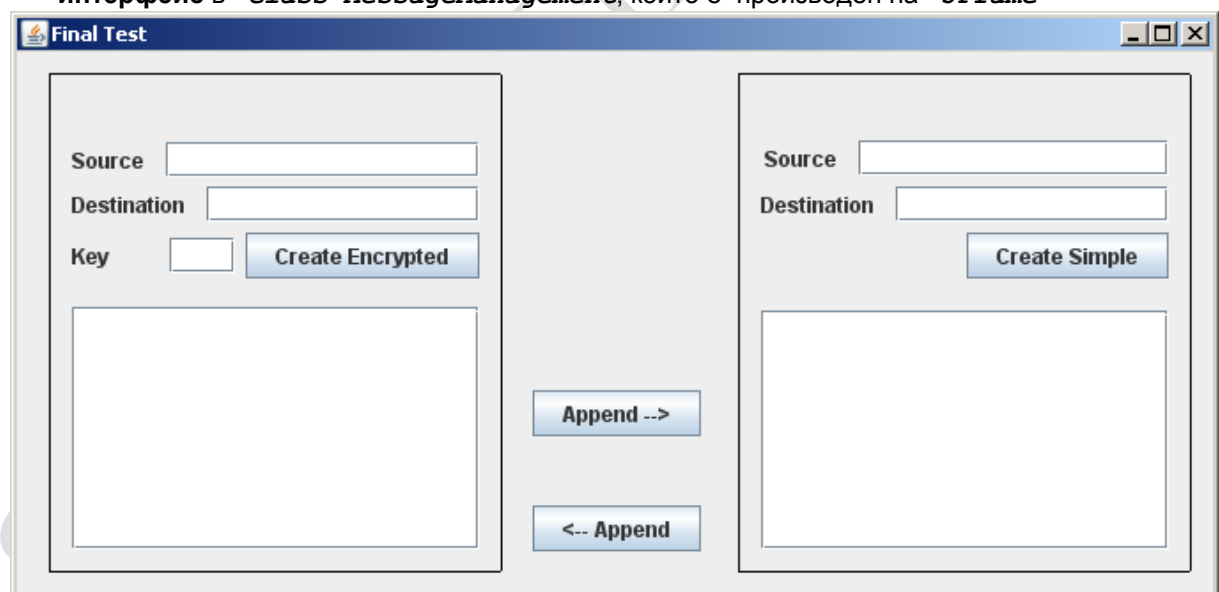
{

//2) долепете отдясно съдържанието на текста в променливите за *source*, *destination* и *txt* на текущия обект към съдържанието на съответните променливи за *source*, *destination* и *txt* на обекта рефериран с *obj* като използвате празен символ за разделител. Текстът *obj* **да се декриптира** преди да се долепи.

}

Точки:10

8. Използвайте **графичния редактор** на *Netbeans* и създайте следния **графичен интерфейс** в *class MessageManagement*, който е произведен на *JFrame*



Точки:10

9. Нека *class MessageManagement* има масив *arrMessageObj* от два елемента от *class Message* .

- При натискане на **бутона** *Create Encrypted* се създава обект от клас *class EncMessage* с данни съответстващи на въведените в текстовите полета на *JPanel*-а отляво на горната картинка . Този обект **да се присвоява** на първия

елемент от масива *arrMessageObj* Нека с **диалогов прозорец** да се изведе **пълна информация за данните** на създадения обект

- При натискане на **бутона *Create Simple*** се създава обект от клас *class TextMessage* с данни съответстващи на въведените в текстовите полета на *JPanel*- а отдясно на горната картинка. Този обект **да се присвоява** на втория елемент от масива *arrMessageObj* Нека с **диалогов прозорец** да се изведе **пълна информация за данните** на създадения обект
- При натискане на **бутона *Append→*** се долепя съобщението (метод *appendTo()*), създадено с **бутона *Create Encrypted*** към обекта на съобщение създадено с **бутона *Create Simple*** Нека с **диалогов прозорец** да се изведе **пълна информация за данните** на така променения обект от *TextMessage*
- При натискане на **бутона *←Append*** се долепя съобщението (метод *appendTo()*), създадено с **бутона *Create Simple*** към обекта на съобщение създадено с **бутона *Create Encrypted***. Нека с **диалогов прозорец** да се изведе **пълна информация за данните** на така променения обект от *EncMessage*

Точки:20