Софийски Университет “Климент Охридски”
Факултет по Математика и Информатика

Контролно No. 1

Курс: **Приложно Обектно Ориентирано Програмиране 1**
Преподавател:проф. д-р. Е. Кръстев **Студент** :
Дата: април 2022 **Време за работа: 120 min**

Инструкции: Изпълнете следното задание за обектно ориентирано програмиране на Java и предайте решенията на IntelliJ в Мудъл. Пълен набор от точки се присъжда за пълно решение на съответната подзадача. Програмната реализация трябва да спазва концепциите за обектно ориентирано програмиране- encapsulation, information hiding, inheritance, polymorphism и избягване на дублиране на код.

Оценки:

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**

Задача 1 (100 точки)

Задание: Играта Bezique се играе от двама с тесте от 64 карти. Всяка карта има цвят и сила. Силите на картите са осем, подредени в намаляващ ред на стойностите им като
ACE, KING, QUEEN, JACK, TEN, NINE, EIGHT, SEVEN, а цветовете са
CLUBS, DIAMONDS, HEARTS, SPADES. От всеки цвят има 16 карти, съставени от две поредици, всяка от които има по една карта от 8-те различни сили. Картите се разбъркват, делят се на две тестета и се теглят последователно извадки от по 12 карти като се започва от тестето с по- малко карти. (в последната извадка има по- малко от 12 карти). Когато картите в първото тесте се изчерпят, картите се теглят от второто тесте. Играчите натрупват точки в зависимост от поредността на силите и цветовете на раздадените им извадки от карти .

A. **Напишете** Java **приложение** на IntelliJ, **наречено** `CardsGameLib`, където **създайте package** `model` **със следното съдържание :**

1. **Напишете клас** ***`Card`***, който **описва всяка една карта от тестето от 64 карти. Добавете в** клас ***`Card`*** следните данни

   ```
   int face; // сила на карта
   int suit; // цвят на карта
   ```

   Добавете публични **статични** едномерни масиви ***`faces`*** и ***`suits`***,които съдържат еднократно съответно **всички различни сили** (8) и **цветове**(4), описани в Заданието и представени с текстовите им описания в това Задание. **Напишете*:***

   - **`get`** и **`set`** **методи** за данните `face` и `suit`
   - **Конструктор за общо ползване**
   - **Метод `String getSuitName()`,** който връща наименованието на **цвета** на текущо създаваната карта, например, ***`CLUBS`***

- **Метод `toString()`,** който връща `String` с името на **силата** и **цвета** на текущо създаваната карта, например, ***KING*** **`of`** ***HEARTS***

Точки:10

2. **Напишете** клас ***`StackOfCards`***, който **описва тесте от 64 карти и дефинира основни методи за игра с картите. Добавете в** клас ***`StackOfCards`*** следните данни:

```
Card[] cards; // масив от всички карти в тестето от карти
Card[] pack1; // масив на първото тесте след деленето на картите
Card[] pack2; // масив на второто тесте след деленето на картите
Card[] hand;  // текущо изтеглена ръка от най- много 12 карти
int trump;   // индекс на цвят в Cards.suits,  Коз на играта
Random random; // генератор на случайни числа
int currentCard; // брой карти изтеглени текущо от cards
int currentPack1; // брой карти изтеглени текущо от pack2
int currentPack2; // брой карти изтеглени текущо от pack1
```

Добавете конструктор по подразбиране, <u>където</u> `cards` се инициализира с <u>всички</u> 64 карти(*четири цвята, всеки с по 8 двойки различни сили***), `trump`** е произволно избран индекс [0,3], а останалите с подразбиращи се стойности да са в съответствие с поставените задачи по- долу**. Добавете също `getTrump()` метод.**

Точки:10

3. **В** клас ***`StackOfCards`*** **напишете следните методи**

a) **Добавете метод**

`public void shuffleCards(),`

който разбърква елементите на масива `cards` в случаен ред и инициализира на нула `currentCard`.

Точки:8

b) **Добавете метод**

`String printCards(),`

който връща текстовото описание на картите в масива `hand`. Картите да се описват по четири на ред (последният ред може да има по- малко) и текстовото им описание да е разделено със запетаи.(*вижте примера в края на текста***) .**

Точки:8

c) **Добавете метод**

`void make2packs(),`

който изпълнява `shuffleCards()` и записва в масива `pack1` първите `N` карти от `cards`, а останалите карти записва в `pack2`. Числото `N` изберете по произволен начин в интервала [16, 48].

Точки:8

d) **Добавете метод**

`boolean dealHand( ),`

който връща `true` или `false` в зависимост дали са останали или не карти за изтегляне в `pack1` и `pack2`. Ако има карти за изтегляне, те се записват последователно в `hand` като се започва от `pack1` и се продължава с изтегляне на карти от `pack2`, когато картите в `pack1` не са останали карти за изтегляне. В `hand` се записват 12 карти или по- малко, ако толкова са останали в `pack2`.

Точки:8

e) **Добавете метод**

`boolean hasKQ,`

който връща `true` или `false` в зависимост дали измежду елементите на `hand` има двойка *KING* и *QUEEN* от някакъв `Suit`.

Точки:5

f) **Добавете метод**

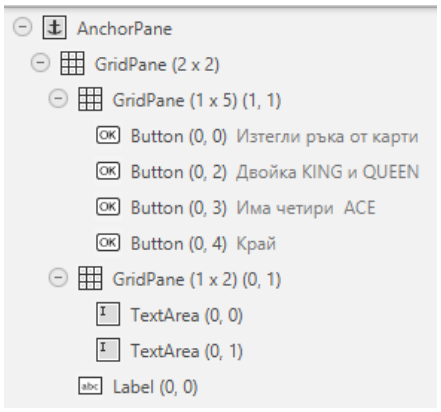
`boolean has4ACE()`

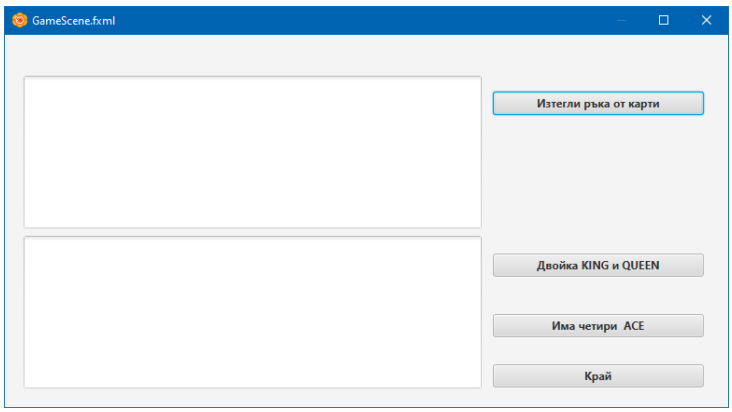
който връща `true` или `false` в зависимост дали измежду елементите на `hand` има четири *ACE* от някакъв `Suit`.

Точки:5

B. Създайте потребителски пакет в JAR формат от проекта `CardGameLib` и напишете нов IntelliJ проект с JavaFX приложение `GameTest`, което да използва този потребителски пакет. В така създаденото конзолно приложение създайте `package gui` със следното съдържание (Точки 8):

1. Добавете към проекта FXML **файл за описание на JavaFX сцена**, съответни **класове** за **Контролер** и **JavaFX приложение** (Application). Създайте сцената като използвате **следната йерархична структура**





Точки:14

2. При инициализацията на Контролера създайте обект от клас ***`StackOfCards`*** и изпълнете метода **`make2packs`**(). В етикета над текстовите области изведете текущо избрания Коз.

Точки: 4

3. Напишете методи за обработка на събитията при натискане на бутоните:
   - При натискане на бутона **Изтегли ръка от карти** изпълнете метода **`dealHand()` на *`StackOfCards`*** и изведете в горната текстова област резултата от метода **`printCards`**(). При липса на карти, бутонът да се деактивира.
   - При натискане на бутона **Има двойка *KING* и *QUEEN*** изпълнете метода **`hasKQ()` на *`StackOfCards`*** и изведете текстово описание на резултата в долната текстова област, както е показано на примера.
   - При натискане на бутона **Има четири *ACE*** изпълнете метода **`has4ACE()` на *`StackOfCards`*** и изведете текстово описание на резултата в долната текстова област, както е показано на примера.
   - При натискане на бутона **Край** изпълнете команда за прекратяване на JavaFX приложението.

Точки:12

Примерна реализация