ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ
Одсек за ИР Београд 25.04.2015.

Колоквијум из Објектно оријентисаног програмирања II

1) *(30 поена)* Одговорити концизно (по једна или две реченице) и прецизно на следећа питања:

а) (1) Којим члановима обухватајућег пакета имају право приступа класе из угнежђених пакета? (2) Којим члановима угнежђених пакета имају право приступа класе из обухватајућег пакета?

б) Да ли се приликом клонирања објекта позива конструктор новог објекта и зашто?

в) Да ли је дозвољено да се: (1) интерфејс изведе из више интерфејса, (2) апстрактна класа изведе из више апстрактних класа, (3) апстрактна класа имплементира више интерфејса?

2) *(укупно 70 поена)* Написати на језику *Java* следећи пакет типова (грешке пријављивати изузецима опремљеним текстовима порука):

- *(25 поена)* ***Алгоритам*** може да шифрује задати карактер који мора бити мало слово енглеске абецеде (грешка је ако карактер не представља мало слово.). Резултат операције је шифровани карактер.
- ***Цезаров*** алгоритам има задати померај у опсегу од 1 до 25 (грешка је ако је померај ван тог опсега). Шифровање карактера се реализује додавањем помераја *p* на задати карактер, при чему се примењује аритметика по модулу 26 (на пример, за *p*=1, `'z'=>'a'`). Може да се састави текстуални опис према следећем формату: **`Cezar:`** *p*`.`
- ***Affine*** алгоритам има задате целобројне кључеве за шифровање *a* и *b* (грешка је ако су вредности кључева негативне). Шифровање карактера се реализује применом функције *y* = `(`*a*`*`*x* `+` *b*`)` `mod 26`, где *x* и *y* представљају редни број карактера, почевши од `0` за `'a'` (на пример, за *a*=5 и *b*=8, `'z'=>'d'`). Може да се састави текстуални опис према следећем формату: **`Affine:`** *a*`,` *b*`.`
- *(45 поена)* ***Цилиндар*** садржи задати алгоритам за шифровање и низ целобројних вредности од 26 елемената. Елементи низа се при стварању цилиндра постављају на шифроване вредности малих слова енглеске абецеде, тако да се на позицији са индексом *i* налази шифрована вредност малог слова са редним бројем *i* (почевши од 0 за 'a'). Цилиндар може да шифрује задати карактер и да се копира, при чему се копира низ, али не и садржани алгоритам. Може да се састави текстуални опис цилиндра који у првом реду садржи текстуални опис садржаног алгоритма, а у наредном реду парове малих слова енглеске абецеде и одговарајућих вредности из низа шифара, за свако слово енглеске абецеде по један пар.
- ***Ротор*** садржи произвољан број цилиндара и ствара се са једним цилиндром који му припада, након чега се цилиндри могу појединачно додавати. Може да се састави текстуални опис који се састоји од описа садржаних цилиндра. Ротор може да шифрује поруку која се састоји од низа малих слова, тако што шифрује сва слова поруке појединачно. При шифровању, свако слово поруке "пролази" кроз све цилиндре на следећи начин: задато слово се шифрује на цилиндру и тако добија шифровано слово, које се задаје следећем цилиндру да га шифрује итд. Шифровање једног слова се завршава на последњем цилиндру ротора, који шифровањем враћа слово које ће се налазити у шифрованој поруци.

Написати на језику *Java* **програм** у којем се ствара неколико алгоритама, цилиндара и ротор коме се додају створени цилиндри и копије цилиндара. Након тога се захтева шифровање поруке од ротора, после чега се ротор исписује на стандардном излазу, као и оригинална и шифрована порука. При писању програма користити константне параметре (не треба ништа учитавати).

НАПОМЕНЕ: **а)** Колоквијум траје **120** минута.

б) Рад се предаје искључиво у вежбанци за испите (-5 поена за неадекватну вежбанку). Није дозвољено имати поред себе друге листове папира, нити уз себе имати мобилни телефон, без обзира да ли је укључен или искључен.

в) Водити рачуна о уредности. Нечитки делови текста ће бити третирани као непостојећи. Решења задатака навести по горњем редоследу (-1 поен за лош редослед). Препоручује се рад обичном графитном оловком.

г) Резултати колоквијума биће објављени на *Web*-у на адреси: `http://rti.etf.bg.ac.rs/rti/ir2oo2/index.html/`

```java
//Algoritam.java
package kriptografija;
import greske.GOpseg;
public abstract class Algoritam {
  public char kriptuj(char c) throws GOpseg {
    if ((c<'a')||(c>'z')) throw new GOpseg();
    return radi(c);
  }
  protected abstract char radi(char c);
}
```

```java
//Cezar.java
package kriptografija;
import greske.*;
public class Cezar extends Algoritam {
  private int pomeraj;
  public Cezar(int p) throws GOpseg {
    if (p < 1 || p > 25) throw new  GOpseg();
    pomeraj = p;
  }
  protected char radi(char c)   {
    return (char) ((c-'a'+pomeraj)%26+'a');
  }
  public String toString() {
    return "Cezar: " + pomeraj;
  }
}
```

```java
//Affine.java
package kriptografija;
import greske.GOpseg;
public class Affine extends Algoritam{
  private int a, b;
  public Affine(int aa,int bb) throws GOpseg {
    if (aa<0 || bb<0) throw new GOpseg();
    a=aa; b=bb;
  }
  protected char radi(char c)  {
    return (char)((a*(c-'a')+b)%26+'a');
  }
  public String toString() {
    return "Affine: " + a + ", " + b;
  }
}
```

```java
//Cilindar.java
package kriptografija;
import greske.GOpseg;
public class Cilindar implements Cloneable {
  private Algoritam alg; private int niz[];
  public Cilindar(Algoritam a) {
    alg = a; niz = new int[26]; popuni();
  }
  public Cilindar clone()
        throws CloneNotSupportedException {
    Cilindar c = (Cilindar) super.clone();
    c.niz = niz.clone(); return c;
  }
  private void popuni() {
    for(int i=0;i<26;i++) {
      try {
        char k=alg.kriptuj((char)('a'+i));
        niz[i]=k-'a';
       } catch (GOpseg e) {
          System.out.println(e);
        }
    }
  }
  public char sifruj(char c) {
    return (char) (niz[c-'a']+'a');
  }
  public String toString() {
    String ret = "Algoritam: " + alg + "\n";
    for (int i=0; i<26; i++)
      ret+=(char)('a'+i)+" "+niz[i]+" ";
    return ret;
  }
}
```

```java
//Rotor.java
package kriptografija;
public class Rotor {
  private Elem prvi, posl;
  private static class Elem {
    Cilindar cln;
    Elem sled;
    Elem(Cilindar c) { cln = c; }
  }
  public Rotor(Cilindar c) { dodaj(c); }
  public Rotor dodaj(Cilindar c) {
    Elem novi = new Elem(c);
    if (prvi == null) prvi = novi;
    else posl.sled = novi; posl = novi;
    return this;
  }
  public String kriptuj(String s) {
    String ret = "";
    char org, sif;
    for (int i = 0; i < s.length(); i++) {
      org = s.charAt(i);
      for(Elem tek=prvi; tek!=null;
                              tek=tek.sled) {
        sif = tek.cln.sifruj(org);
        org = sif;
      }
      ret += sif;
    }
    return ret;
  }
  public String toString() {
    String s = "";
    for (Elem tek=prvi;tek!=null;tek=tek.sled)
      s += tek.cln + "\n";
    return s;
  }
}
```

```java
// Main.java
import kriptografija.*;
public class Main {
  public static void main(String[] args) {
    try {
      Algoritam a1=new Cezar(3),
           a2=new Affine(5, 8);
      Cilindar c1=new Cilindar(a1),
           c2=new Cilindar(a2),c3=c1.clone();
      Rotor r = new Rotor(c1);
      r.dodaj(c2).dodaj(c3);
      String original = "kolokvijum";
      String kriptovano=r.kriptuj(original);
      System.out.println(r+"\n"+original+" - "
           +kriptovano);
    } catch (Exception e) {
        System.out.println(e);
      }
  }
}
```

```
// primer izvršavanja

Algoritam: Cezar: 3
a 3 b 4 c 5 d 6 e 7 f 8 g 9 h 10 i 11 j 12 k
13 l 14 m 15 n 16 o 17 p 18 q 19 r 20 s 21 t
22 u 23 v 24 w 25 x 0 y 1 z 2
Algoritam: Affine: 5, 8
a 8 b 13 c 18 d 23 e 2 f 7 g 12 h 17 i 22 j 1
k 6 l 11 m 16 n 21 o 0 p 5 q 10 r 15 s 20 t
25 u 4 v 9 w 14 x 19 y 24 z 3
Algoritam: Cezar: 3
a 3 b 4 c 5 d 6 e 7 f 8 g 9 h 10 i 11 j 12 k
13 l 14 m 15 n 16 o 17 p 18 q 19 r 20 s 21 t
22 u 23 v 24 w 25 x 0 y 1 z 2

kolokvijum - ysdsybotwi
```