NTP 2020/2021.

Vežbe 5. i 6.

U četvrtom i petom terminu vežbi studenti bi trebalo da se upoznaju sa *Pharo* okruženjem, samostalno proučavajući dolenavedene materijale:

- *Pharo* prezentacija
- *Pharo* video-predavanje 1.
- *Pharo* video-predavanje 2.
- *Pharo MOOC*:
  - prve četiri nedelje (*week 1-4*)
  - ignorisati materijali označene tagovima *web* i *TinyBlog*

Zadaci za samostalni rad

Nakon prolaska kroz gorenavedene materijale, ali i uz dodatno samostalno istraživanje, potrebno je uraditi sledeće zadatke (teorijske i praktične):

Pitanja koja se odnose na oblast objektno orijentisano programiranje:

1. Šta je objektno orijentisano programiranje (skr. OOP)?
2. Šta je klasa?
3. Šta je objekat?
4. Da li je postajanje objekata uslovljeno postojanjem klasa? Objasniti.
5. Šta je enkapsulacija?
6. Šta je polimorfizam?
7. Šta je interfejs (engl. *interface*)?
8. Šta je apstraktna klasa?
9. Koja je razlika između interfejsa i apstraktne klase?
10. Objasniti pojam nasleđivanja. Kakvo sve nasleđivanje može biti?
11. Kako se definiše stanje, a kako ponašanje objekata?
12. Kako se u kontekstu OOP-a iskazuje izbor?

Pitanja koja se odnose na *Pharo* okruženje:

13. Da li je *Pharo* statički ili dinamički tipiziran programski jezik? Objasniti.
14. Da li u programskom jeziku *Pharo* postoje tipovi?
15. Da li su klase objekti?
16. Da li su poruke objekti?
17. U čemu je razlika između poruke i metode? Šta se iskazuje porukom, a šta metodom?
18. Kako su modifikatori pristupa realizovani u programskom jeziku *Pharo*?
19. Kako se u programskom jeziku *Pharo* definiše statička metoda?
20. Kako se u programskom jeziku *Pharo* definiše interfejs?
21. Kako se u programskom jeziku *Pharo* definišu parametrizovani tipovi, odnosno generici (engl. *generics*)?
22. Kojoj klasi pripada `nil` u programskom jeziku *Pharo*? Koliko instanci ovog objekta postoji?
23. Kakve sve poruke mogu biti? Navesti primere za svaku kategoriju poruka.

24. Kako su u programskom jeziku *Pharo* definisani prioriteti poruka?
25. Kako su u programskom jeziku *Pharo* definisani prioriteti matematičkih operatora?
26. Šta je objekat prijemnik (engl. *receiver*), a šta selektor poruke (engl. *selector*)?
27. Koje tri komponente može da sadrži poruka?
28. Šta predstavlja znak **;** u programskom jeziku *Pharo* ?
29. Šta predstavlja znak **.** u programskom jeziku *Pharo*?
30. Koja je razlika između **;** i **.** u programskom jeziku *Pharo*?
31. Šta je kaskada? Šta je rezultat kaskade? Šta je rezultat višestruke upotrebe **;** ?
32. Da li je izraz `1 * 2; + 100; * 123.` sintaksno validan? Ukoliko jeste, odrediti njegov rezultat, u suprotnom objasniti grešku koja se javlja prilikom parsiranja.
33. Šta su simboli u programskom jeziku *Pharo*? Navesti nekoliko primera.
34. Koji sve literali postoje u programskom jeziku *Pharo*? Navesti primere.
35. Šta su blokovi? Da li su blokovi objekti?
36. Šta je vrednost definicije, a šta evaluacije bloka?
37. Da li blokovi mogu da primaju argumente? Ukoliko mogu, kako im se argumenti prosleđuju? Navesti primere.
38. Da li se blok može smestiti u privremenu promenljivu? Da li se blok može proslediti kao argument poruke? Da li blok može biti povratna vrednost metode?
39. Koliko puta se jedan blok može evaluirati?
40. Šta će se desiti ukoliko se prilikom evaluacije bloka naiđe na znak **^** ?
41. Šta je `yourself`? Kada se `yourself` najčešće upotrebljava? Navesti primer.
42. Kakve sve promenljive mogu biti u programskom jeziku *Pharo*? Navesti primere.
43. Šta su globalne, a šta lokalne promenljive u programskom jeziku *Pharo*? Navesti primere.
44. Kako su petlje implementirane u programskom jeziku *Pharo*?
45. Šta su kolekcije u programskom jeziku *Pharo*? Navesti nekoliko primera najčešće korišćenih kolekcija.
46. Koja je razlika između niza (`Array`) i uređene kolekcije (`OrderedCollection`)?
47. Šta su literalni nizovi? Koje sve elemente mogu da sadrže?
48. Kada se kreiraju literalni nizovi?
49. Šta su iteratori? Kako su implementirani u programskom jeziku *Pharo*?
50. Na koji način iteratori doprinose enkapsulaciji kolekcije?
51. Objasniti šta znači da se iteratori ponašaju polimorfno u programskom jeziku *Pharo*?
52. Koje su *Pharo* poruke ekvivalenti funkcijama `map`, `filter` i `reduce` programskog jezika *Python*?
53. Šta je `true`, a šta `false` u programskom jeziku *Pharo*? Kojim klasama pripadaju? Koliko instanci ovih objekata postoji?
54. U čemu je razlika između poruka & i `and:` ?
55. Kako se iskazuju uslovi u programskom jeziku *Pharo*?
56. Objasniti pojam toka (engl. s*tream*). Sa čime sve tokovi mogu da manipulišu?
57. Šta je podrazumevana povratna vrednost metode?
58. Šta znači da su metode virtualne (engl. *virtual*)?
59. Šta znači da se metode kasno vezuju (engl. *late binding*)?
60. Zašto se u kontekstu programskog jezika *Pharo* koristi pojam ***slanje poruke*** (engl. *message passing*), a ne pojam *poziv metode*?
61. Kako se u programskom jeziku *Pharo* kreira klasa?
62. Kako se u programskom jeziku *Pharo* implementira konstruktor klase?
63. Kako se u programskom jeziku *Pharo* kreiraju objekti? Navesti primere.
64. Kakvo nasleđivanje je podržano u programskom jeziku *Pharo*?

65. Koja je korenska klasa u hijerarhiji nasleđivanja u programskom jeziku *Pharo*?
66. Kako se i kada nasleđuju stanje, odnosno ponašanje objekata u programskom jeziku *Pharo*?
67. Iz koja dva koraka se sastoji slanje poruke?
68. Na šta pokazuje *self*, a na šta *super*?
69. U čemu je razlika između *self* i *super*?
70. Kako se i kada određuju *self*, odnosno *super*?
71. Koja je uloga pojma ***pretraga metoda*** (engl. m*ethod look-up*)?
72. Iz kojih koraka se sastoji pretraga metoda (engl. *method look-up*)?
73. Šta se desi kada pretraga metoda (engl. *method look-up*) ne uspe?
74. Kako se u programskom jeziku *Pharo* navodi par (*ključ:vrednost*)?

Praktični zadaci – programski jezik Pharo:

75. Napisati beskonačnu petlju u programskom jeziku *Pharo*[1].

76. Na koje se literale programskog jezika *Pharo* odnose sledeći iskazi?

- `'Hello', 'Dave'`
- `1.3`
- `#node1`
- `#(2 33 4)`
- `[ :each | each scale: 1.5 ]`
- `$A`
- `true`
- `1`
- `"Hello, world!"`

77. Za svaki od dolenavedenih iskaza odrediti:
    (a) objekat prijemnik poruke,
    (b) selektor poruke,
    (c) argumente poruke,
    (d) rezultat izvršavanja iskaza.

    - `3 + 4`
    - `Date today`
    - `anArray at: 1 put: 'hello'`
    - `anArray at: i`
    - `#(2 33 -4 67) collect: [ :each | each abs ]`
    - `25 @ 50`
    - `SmallInteger maxVal`
    - `#(a b c d e f) includesAll: #(f d b)`
    - `true | false`
    - `Point selectors`

1 Voditi računa da prilikom izvršavanja beskonačne petlje može doći do zamrzavanja *Pharo* virtualne mašine.

78. Odrediti rezultat izvršanja sledećih iskaza.

- ```
  | anArray |
  anArray := #('first' 'second' 'third' 'fourth').
  anArray at: 2
  ```
- ```
  #(2 3 -10 3) collect: [ :each | each * each]
  ```
- ```
  6 + 4 / 2
  ```
- ```
  1 + 3 negated
  ```
- ```
  1 + (3 negated)
  ```
- ```
  2 raisedTo: 3 + 2
  ```
- ```
  2 negated raisedTo: 3 + 2
  ```
- ```
  #(a b c d e f) includesAll: #(f d g)
  ```

79. Napisati ekvivalentne iskaze upotrebom najmanjeg mogućeg broja zagrada.

- ```
  ((3 + 4) + (2 * 2) + (2 * 3))
  ```
- ```
  x between: (pt1 x) and: (pt2 y)
  ```
- ```
  (x isZero)
     ifTrue: [....].
  (x includes: y)
     ifTrue: [....]
  ```
- ```
  (OrderedCollection new)
     add: 56;
     add: 33;
     yourself
  ```
- ```
  (Integer primesUpTo: 64) sum
  ```
- ```
  ('http://www.pharo.org' asUrl) retrieveContents
  ```
- ```
  (('2014-07-01' asDate) - '2013/2/1' asDate) days
  ```
- ```
  (((ZnEasy getPng: 'http://pharo.org/web/files/pharo.png')
     asMorph) openInWindow)
  ```
- ```
  ((#(a b c d e f) asSet) intersection: (#(f d b) asSet))
  ```
  ```

80. Za sledeće iskaze ispisati redosled slanja poruka, odnosno izvršavanja odgovarajućih metoda.

- `Date today daysInMonth`
- `5@5 extent: 6.2 truncated @ 7`
- `Transcript show: (45 + 9) printString`
- `('2014-07-01' asDate - '2013/2/1' asDate) days`
- `42 factorial decimalDigitLength`
- ```
  (ZnServer startDefaultOn: 8080)
     onRequestRespond: [ :request | ZnResponse ok: (ZnEntity
     with: DateAndTime now printString) ]
  ```
- `(1914 to: 1945) count: [ :each | Year isLeapYear: each ].`
- `$/ join: ($- split: '1969-07-20') reverse`
- ```
  DateAndTime fromUnixTime:
     ((ByteArray readHexFrom: 'CAFEBABE4422334400FF')
        copyFrom: 5 to: 8) asInteger
  ```
- `(String new: 32) collect: [ :each | 'abcdef' atRandom ]`
- `'http://www.pharo.org' asUrl saveContentsToFile: 'page.html'`
- ```
  '^.*.jpg' asRegex in: [ :regex |
     '/tmp/foo.txt' asFileReference contents lines
        select: [ :line | regex matches: line ] ]
  ```

81. Dat je sledeći isečak koda u programskom jeziku *Pharo*.

```
1
2  | sum |
3  sum := 0.
4  #(21 23 53 66 87) do: [:item | sum := sum + item].
5  sum
6
```

(a) Šta će biti vrednost promenljive `sum` nakon izvršavanje gorenavedenog isečka koda?
(b) Napisati ekvivalentni kod tako da koristi eksplicitno indeksiranje elemenata upotrebom poruke `at:` .
(c) Napisati ekvivalentni kod tako da koristi poruku `inject:into:` .

82. Upotrebom `Zinc` *HTTP* biblioteke dostupne u *Pharo* okruženju, potrebno je dobaviti logo sa sledeće veb-adrese: http://pharo.org/files/pharo.png. Dobavljeni sadržaj pretvoriti u `morph` grafički objekat i prikazati na ekranu.
Pomoć:
    - Klasa `ZnEasy` ima dosta korisnih poruka za manipulisanje *HTTP* zahtevima.
    - Zadatak poruke `asMorph` je da objekat prijemnik pretvori u `Morph` grafički objekat.
    - Jedan od načina kako se grafički objekat može prikazati na ekranu jeste slanjem poruke `openInWorld`.
```

83. Upotrebom `Zinc` *HTTP* biblioteke preuzeti sliku vezanu za e-mail adresu *stephane.ducasse@inria.fr* putem gravatara (http://www.gravatar.com/avatar/). Sliku pretvoriti u `morph` grafički objekat i prikazati na ekranu.
    Pomoć:
    - Putanja do slike vezane za e-mail adresu ima sledeći format: `http://www.gravatar.com/avatar/<MD5_heksadecimalni_šifat_e-maila>.jpg`.
    - `MD5 hashMessage:` poruka je zadužena za pravljenje šifrata, koji kasnije treba konvertovati u heksadecimalni oblik.
    - Stringovi ne smeju sadržati velika slova i ne smeju imati beline (engl. *whitespaces*) na početku, odnosno na kraju.
    - Pogledati način upotrebe poruka `openInHand` i `asMorph`.

84. U jednom iskazu potrebno je odrediti zbir kvadrata prvih 100 prirodnih brojeva.
    Napomena: Nije dozvoljena upotreba poruke `sum`.

85. U jednom iskazu potrebno je odrediti zbir prvih 100 parnih prirodnih brojeva.
    Napomena: Nije dozvoljena upotreba poruke `sum`.

86. Napisati blok (anonimnu funkciju) koji kao argument prima jedan prirodan broj. Zadatak bloka je da u jednom iskazu odredi (desni) faktorijel za prosleđeni parametar, tj. ***n!***[2].
    Napomena: Nije dozvoljena upotreba poruke `factorial`.

87. Napisati blok (anonimnu funkciju) koji kao argument prima jedan prirodan broj. Zadatak bloka je da u jednom iskazu odredi *levi faktorijel* za prosleđeni argument, tj. ***!n***[3]. Potrebno je iskoristiti blok iz prethodnog zadatka.
    Napomena: Nije dozvoljena upotreba poruka `sum`, odnosno `factorial`.

88. Data je uređena kolekcija `#(1 2 3 4 5 6 7 8 9 10)`. Upotrebom jednog iskaza odrediti zbir parnih, odnosno neparnih brojeva. Rezultat iskaza treba da bude uređena kolekcija koja sadrži dva broja. Prvi broj predstavlja zbir parnih brojeva, dok drugi broj predstavlja zbir neparnih brojeva.
    Napomena: Nije dozvoljena upotreba poruke `sum`.
    Pomoć: Upotrebiti poruku `groupedBy:`.

Napomena: Zadatke 83-87. rešiti u maniru funkcionalnog programiranja.

2 https://en.wikipedia.org/wiki/Factorial
3 https://sh.wikipedia.org/wiki/Levi_faktorijel