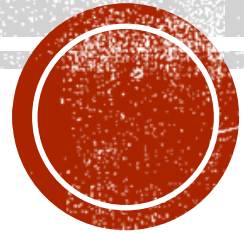
ОБЈЕКТНО ОРИЈЕНТИСАНО ПРОГРАМИРАЊЕ

ПРОГРАМСКИ ЈЕЗИК ЈАВА – 1

Угњеждене и унутрашње класе



УГЊЕЖДЕНЕ КЛАСЕ

- Јава допушта да се дефинише класа унутар неке друге класе.
  - Таква класа се назива угњеждена класа.
- Постоје два типа угњеждених класа:

```
class SpoljasnjaKlasa{
    ...
    static class StatickaUgnjezdenaKlasa{
      ...
    }
    ...
    class UnutrasnjaKlasa{
      ...
    }
}
```

УГЊЕЖДЕНЕ КЛАСЕ (2)

- Разлика је у постојању модификатора **static.**
  - Нестатичка угњеждена класа се заправо зове унутрашња класа с обзиром на њену блиску везу са спољашњом класом.
  - Она се може сматрати њеном правом чланицом попут нестатичких метода и нестатичких поља.
- Постоји неколико разлога за коришћење угњеждених класа.
  - То је начин логичког груписања класа које се користе само на једном месту.
  - Тиме се побољшава енкапулација (учаурење).
  - Ако се класа B смести унутар класе A,
    из метода класе B се може приступати пољима класе A чак иако су она приватна.
  - Угњеждене класе доводе до читљивијег кода који се лакше одржава.
  - Угњеждавање малих класа у велику доводи до тога
    да код неке операције буде смештен близу места коришћења.

СТАТИЧКЕ УГЊЕЖДЕНЕ КЛАСЕ

- Статичка угњеждена класа се може упоредити са статичким пољем или статичком методом.
- Веза са спољашњом класом је “лабава” мотивисана могућношћу погодне логичке организације кода.
  - Слично идеји пакета само ово омогућава груписање на још нижем нивоу гранулације.
- Животни циклус инстанци статичке угњеждене класе није зависан од животног циклуса објеката спољашње класе.

```
StatickaUgnjezdenaKlasa staticka = new Spoljasnja.StatickaUgnjezdenaKlasa();
```

- Дакле, нигде се не спомиње инстанца спољашње класе.

СТАТИЧКЕ УГЊЕЖЂДЕНЕ КЛАСЕ (2)

- Могуће је креирати велики број инстанци статичке угњеждене класе.
  - Оне ће све бити независни објекти на хипу.

```
StatickaUgnjezdenaKlasa staticka1 = new Spoljasnja.StatickaUgnjezdenaKlasa();
StatickaUgnjezdenaKlasa staticka2 = new Spoljasnja.StatickaUgnjezdenaKlasa();
SpoljasnjaKlasa spoljasnja1= new Spoljasnja();
SpoljasnjaKlasa spoljasnja2= new Spoljasnja();
```

- Статичка угњеждена класа не може приступати нестатичким пољима спољашње.
  - Јер инстанца спољашње класе уопште не мора да постоји (као статичка поља и методи).
  - А чак и да постоји, опет не би било јасно на коју се инстанцу спољашње класе мисли.
  - Исто важи и за приступ нестатичким методима спољашње класе.
- Са друге стране, статичка угњеждена класа има приступ статичким пољима и методима спољашње класе, чак иако су та поља приватна.

ПРИМЕР 1

- Написати Јава програм који рачуна плату на месечном нивоу.
- Плата је дефинисана као умножак броја радних сати и цене сата.
- Минималан број сати које радник мора да оствари је 160 (норма), а све преко тога се рачуна као прековремени рад.
- Прековремени сати су реткост и постоји засебан механизам рачунања прековременог додатка.
  - За сваки наредни сат преко норме увећава се цена по сату за 2%.
  - На пример, ако је радник радио 1 прековремени сат добиће 102% цене сата као додатак. Ако је радио два сата, добиће 102%+ 104.04%=206.04% итд.
- Максимална цена по сату је 2000 РСД било да је реч о основној цени рада или прековременој.

ПРИМЕР 1 (2)

- Решење овог задатка је реализовано кроз три класе **ObracunPlata**, **PrekovremeniRad** и **ObracunPlataTest**, где је **PrekovremeniRad** приватна статичка угњеждена класа унутар класе **ObracunPlata.**
- Иако је било могуће **main**() метод убацити у класу **ObracunPlata**, намерно је он записан у одвојеној класи **ObracunPlataTest** како би се видео ефекат модификатора **private** над угњежденом класом.

ПРИМЕР 1 (3)

```
public class ObracunPlata {
    private static int minimalnoSati=160;
    private static int maksimalnaCenaSata=2000;
    private int cenaSata;

    public ObracunPlata(int cenaSata) { … }

    public double izracunajPlatu(int brojSati) {
      …
      double plata = minimalnoSati*cenaSata;
      if(brojSati>minimalnoSati) {
          PrekovremeniRad prekovremeni = new PrekovremeniRad(cenaSata,
                  brojSati-minimalnoSati);
          plata+=prekovremeni.izracunajCenuPrekovremenog();
      }
      return plata;
    }

    private static class PrekovremeniRad{
      private static double koeficijentUvecanja=1.02;
      private int cenaSata;
      private int prekovremenoSati;

      private PrekovremeniRad(int cenaSata, int prekovremenoSati) { … }

      private double izracunajCenuPrekovremenog() { … }
    }
}
```

ПРИМЕР 1 (4)

```
ObracunPlata op1 = new ObracunPlata(1000);
ObracunPlata op2 = new ObracunPlata(1800);
ObracunPlata op3 = new ObracunPlata(800);
ObracunPlata op4 = new ObracunPlata(2200);
System.out.println(String.format("%.2f", op1.izracunajPlatu(162)));
System.out.println(String.format("%.2f", op2.izracunajPlatu(170)));
System.out.println(String.format("%.2f", op3.izracunajPlatu(160)));
System.out.println(String.format("%.2f", op4.izracunajPlatu(162)));

// ObracunPlata.PrekovremeniRad pr

//              =new ObracunPlata.PrekovremeniRad(1000, 23);
// System.out.println(pr.izracunajCenuPrekovremenog());
```

УНУТРАШЊЕ КЛАСЕ

- Унутрашња класа се придружује инстанци класе која је обухвата
  па и она има директан приступ до свих поља и метода објекта који је садржи.
- Како је унутрашња класа придружена инстанци, тј. објекту,
  то она сама не може садржати статички члан.
  - Објекти унутрашње класе постоје само у оквиру примерка спољашње класе.
- Када се дефинише унутрашња класа, она је члан спољашње класе на исти начин као и остали чланови (поља и методи).
- Унутрашња класа може имати модификаторе приступа као и остали чланови.

```
Spoljasnja spoljasnja = new Spoljasnja();
UnutrasnjaKlasa unutrasnja1 = spoljasnja.new Unutrasnja();
UnutrasnjaKlasa unutrasnja2 = spoljasnja.new Unutrasnja();
```

УНУТРАШЊЕ КЛАСЕ (2)

- Синтакса може бити краћа у ситуацијама када се инстанца унутрашње класе креира у нестатичком методу спољашње.
- Тада се може изоставити кључна реч **this** која упућује на инстанцу спољашње.
  - Јасно је да, у случају статичког метода, ово није могуће.

```
class Spoljasnja{
    ...
    void f(){
      ...
      Unutrasnja unutrasnja1 = new Unutrasnja();
      Unutrasnja unutrasnja2 = this.new Unutrasnja();
      ...
    }
    ...
}
```

ПРИМЕР 2

- Написати Јава програм који реализује повезану листу података **PovezanaLista.**
- Податак у сваком елементу листе је типа **Object.**
- Дефинисати и помоћни метод за додавање елемента на крај листе и редефинисати **toString()** за приказ свих елемената листе.
- Онемогућити инстанцирање елемената листе ван тела класе **PovezanaLista.**

ПРИМЕР 2 (2)

- Решење погледати у књизи.

ЛОКАЛНЕ УНУТРАШЊЕ КЛАСЕ

- Поред унутрашње класе, која се сматра чланицом спољне класе, постоје још две варијанте унутрашњих класа.
- У питању су локална унутрашња класа и анонимна класа.
- Локална унутрашња класа није чланица спољашње класе, већ је чланица метода.
  - Због тога се она дефинише и инстанцира у телу припадајућег метода.
  - На овај начин је ниво енкапсулације додатно увећан у односу на обичну унутрашњу класу.
- Локалне унутрашње класе су уведене јер је у неким ситуацијама потребно дефинисати класу за потребе реализације само појединачног метода.
- Локална унутрашња класа може позивати променљиве декларисане у припадајућем методу, под условом да су те променљиве финалне.

ПРИМЕР 3

```
public class ValidacijaBrojaMobilnog {

    static boolean validirajBrojMobilnog(String mobTekst) {
      …
      class BrojTelefona {
          String mobTekst;
          String mobTekstStd;

          BrojTelefona(String mobTekst) {
              this.mobTekst = mobTekst;
              standardizuj();
          }

          void standardizuj() { … }

          String pozivni() { … }

          String broj() { … }
      }
      BrojTelefona brojTelefona = new BrojTelefona(mobTekst);
      if (brojTelefona.mobTekstStd == null)
          return false;
      String pozivni = brojTelefona.pozivni();
      boolean dozvoljen = false;
      …
    }
}
```

АНОНИМНЕ КЛАСЕ

- Анонимне класе су попут локалних унутрашњих класа, само што нису именоване.
- Последица овога је истовремено дефинисање и инстанцирање анонимне класе.
- Инстанцирање се не може извршити више пута, јер не постоји назив класе па самим тим ни назив конструктора.
- Анонимне класе, приликом компилације, добијају аутоматски генерисано име које није од интереса програмеру, али јесте компајлеру.
- Најчешћа намена анонимне класе је пренос функционалности у позив метода.
  - Будући да Јава не подржава рад са показивачима па самим тим ни са показивачима на методе (функције), пренос се реализује слањем објекта који садржи дефиницију метода.
  - Овакви објекти се називају функционали.

ПРИМЕР 4

- Написати Јава програм који сортира низ ниски растуће према суми **ASCII** вредности ниске.
- Ако две ниске имају исту суму **ASCII** вредности, као секундарни критеријум поређења, користити стандардно растуће лексикографско уређење.
- За сортирање користити уграђени метод **Arrays.sort()** и проследити му одговарајући функционал за поређење.

ПРИМЕР 4 (2)

- Решење погледати у књизи.

ПИТАЊА И ЗАДАЦИ

- Шта су угњеждене класе и које су предности употребе угњеждених класа?
- Објаснити концепт статичке угњеждене класе. Којим елементима спољашње класе може да приступа статички угњеждена класа? Примером илустровати употребу статички угњеждене класе.
- Које су разлике између статичке угњеждене класе и унутрашње класе? Илустровати примером.
- Када је корисно дефинисати унутрашњу класу, а када локалну унутрашњу класу? Илустровати примерима.
- Шта су анонимне класе и када се користе?