Објектно орјентисано програмирање

Владимир Филиповић
vladaf@matf.bg.ac.rs
Александар Картељ
kartelj@matf.bg.ac.rs

Модификатори у програмском језику Јава

Владимир Филиповић
vladaf@matf.bg.ac.rs
Александар Картељ
kartelj@matf.bg.ac.rs

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Модификатори

- То су специјалне кључне речи које мењају понашање класа, метода, променљивих или наредби увоза.
- Модификатори су опциони и њихов редослед није битан.
- У неким ситуацијама су имплицитно дефинисани, тј. подразумева се њихово коришћење.
- Постоји велики број модификатора у Јави:
    1. модификатори контроле приступа,
    2. модификатори синхронизације,
    3. нативни модификатори,
    4. модификатори - анотације итд.

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Модификатори за контролу видљивости, тј. приступа

- Постоје 4 нивоа видљивости, тзв. “4P-заштита”.
  (public, package, protected, private)
- Описати повезаност модификатора са концептом учауривања?
- **public** - омогућава видљивост променљиве (или метода у свим класама (чак из различитих пакета).

```
public class SvimaDostupna {
        public int javnaPromenljiva;
        public String niskaJavna;

        public float javniMetod() {
                ……
        }
}
```

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Модификатори за контролу видљивости, тј. приступа (2)

- **package** - служи за сужавање видљивости променљивих и метода на ниво пакета.
- Карактерише га непотребност навођења модификатора. То је подразумевани ниво заштите.

```
public class PodrazumevanaZastita {
        int PaketnaCela = 3;
        String paketnaNiska = "Pera";

        float metodPaketa() {
                ……
        }
}
```

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Модификатори за контролу видљивости, тј. приступа (3)

- protected - ниво видљивости по линији наслеђивања односно на нивоу класе и њених поткласа (директних и индиректних).
- Саопштава да је дозвољено коришћење метода и променљивих само:
  1. од стране класа истог пакета,
  2. или од поткласа изван пакета
     (ако класа у пакету има потклассе изван пакета).
- У пракси се заштићена поља требају пажљиво користити, јер би свака измена довела до проблема у наслеђеним класама.
- Заштићени методи имају више смисла. То индицира да се подкласама може веровати да ће успешно користити метод, а да друге класе то не могу.

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Модификатори за контролу видљивости, тј. приступа (4)

- private - највиши ниво заштите.
- Методи и променљиве са овим модификатором само из класе у којој су дефинисани.

```
public class KlasaSaPrivatnim {
        private int privatnaCela;

        private float privatniMetod() {
                ……
        }
}
```

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Модификатори за контролу видљивости, тј. приступа (5)

- Зашто треба користити приватне променљиве и методе?

```
class Krug {
        int x, y, r;
        ……..

        Krug(int x, int y, int r) {
                …….
        }

        void crtaKrug () {
                …..
        }
}
```

- У овом примеру се може се променити вредност поља за примерке класе Krug из ма које класе која се налази у истом пакету у коме се налази и класа Krug.

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Модификатори за контролу видљивости, тј. приступа (6)

- Овде се користе приватне променљиве примерка, па је за рад са њима потребно позвати методе за очитавање/постављање (енг. getter/setter).

```
class Krug {
        private int x, y, r;
        ……..

        public int getR () {
                return r;
        }

        public int setR(int vred) {
                r = vred;
                crtakrug();
                drugimetod();
                return r;
        }
}
```

Математички факултет
{vladaf,kartelj}@matf.bg.ac.rs

Модификатори за контролу видљивости, тј. приступа (7)

Табела видљивости

Видљивост	public	protected	package	private
1. из исте класе	да	да	да	да
2. из класе у пакету	да	да	да	не
3. из било које класе	да	не	не	не
4. из поткласе ван пакета	да	да	не	не

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Модификатор static

- Овај модификатор мења значење променљивих, метода, наредби увоза и иницијализационих блокова.
- static - модификатор и његово коришћење код:
  1. Променљиве – променљива везана за постојање класе;
  2. Метода – метода везана за постојање класе;
  3. Наредбе увоза – омогућава употребу статичких поља и метода из наведених класе без употребе пуне квалификације.
  4. Иницијализационим блоком – иницијализација статичких поља.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Модификатор final

- Овај модификатор утиче на променљиве, методе и класе модификујући њихово значење на следећи начин:
  1. Класе не могу бити наслеђене;
  2. Поља не могу бити промењена након иницијализације;
  3. Методе не могу бити редефинисане;
  4. Параметри метода не могу бити мењани унутар метода.

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Модификатор abstract

- Користи се за дефинисање апстрактних класа и метода.
- Класа је апстрактна ако се не могу направити конкретни објекти тог типа, већ служи за обезбеђивање информација за потклассе.
- Детаљни опис апстрактних класа је дат у следећој презентацији.

```
public abstract class MojaAps {
        int broj1, broj2;
        ……..

        abstract void f1();
}
```

Математички факултет

{vladaf,kartelj}@matf.bg.ac.rs

Захвалница

Велики део материјала који је укључен у ову презентацију је преузет из презентације коју је раније (у време када је он држао курс Објектно орјентисано програмирање) направио проф. др Душан Тошић.

Хвала проф. Тошићу што се сагласио са укључивањем тог материјала у садашњу презентацији, као и на помоћи коју ми је пружио током конципцирања и реализације курса.