RADNO OKRUŽENJE I API

Radna okruženja i ponovna upotreba koda

- Cilj softverskog inženjerstva je razvoj **robusnog i ponovo upotrebljivog softvera - višekratna upotrebljivost koda** se može **ostvariti** kroz **standardizaciju – koncept "priključi i radi"**
- Koncept **radnog okruženja** (engl. **framework**) se zasniva na principima "priključi i radi" i višekratne upotrebljivosti
- Primeri: Microsoft Office – Word, Excel, PowerPoint imaju većinu zajedničkih stavki u menijima (File, Edit, View, Format...), Windows – rad sa prozorima – radno okruženje sadrži gotove elemente, **ne treba stalno ponovo izmišljati točak**!
- **Programer koristi radno okruženje za pravljenje aplikacija tako što upotrebljava gotove interfejse**

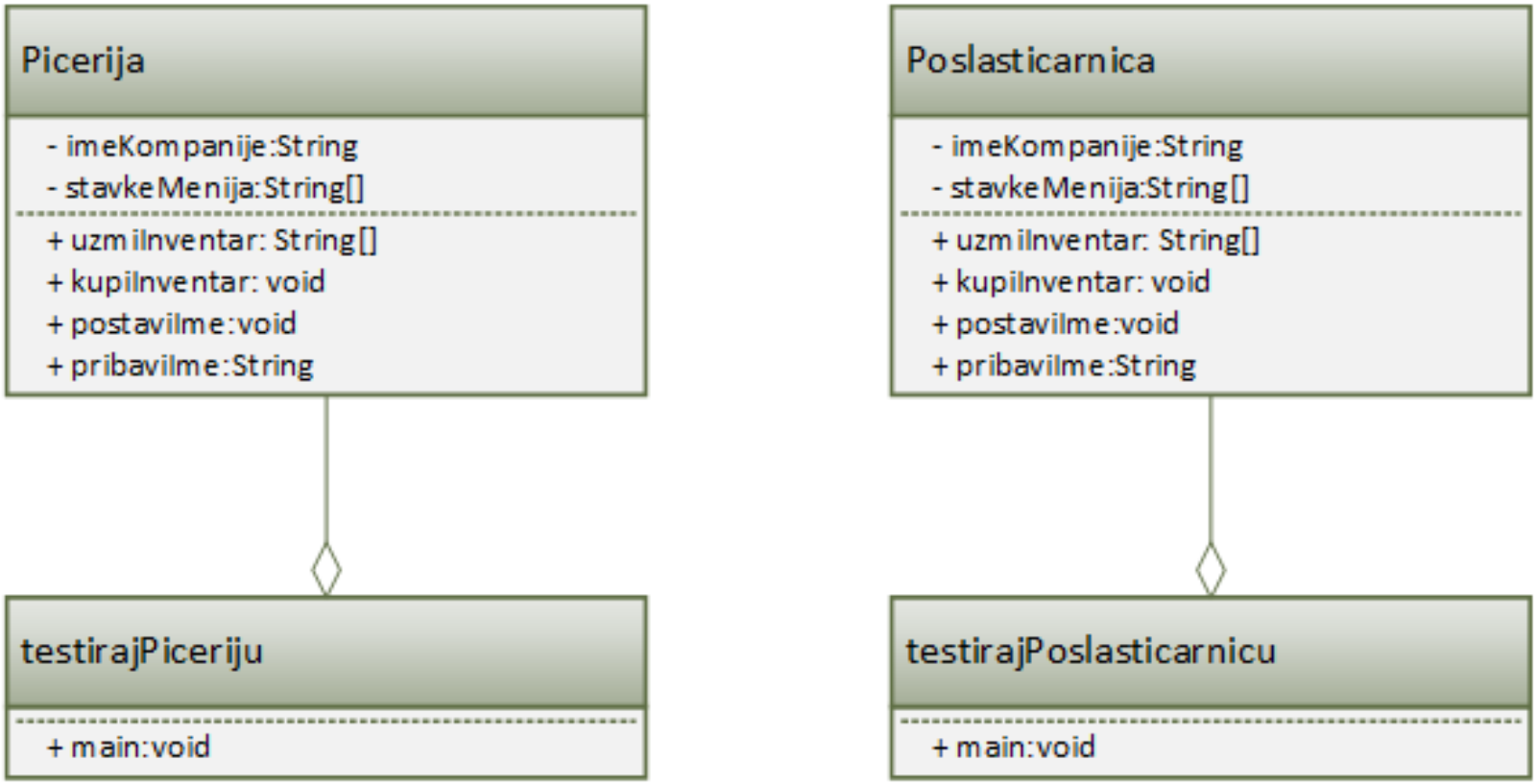
Radna okruženja i API

- **Prednosti: osećaj usaglašenosti i jednoobraznosti, programer može koristiti kod koji je već napisan i testiran**
- Kako se koriste gotovi okviri za dijalog iz radnog okruženja? Postoje pravila koje postavlja radno okruženje, obično organizovana u dokumentaciji napisanoj od strane kreatora okruženja, što dovodi do pojma **programskog interfejsa aplikacije** (engl. **Application Programming Interface – API**)
- Korišćenjem API-ja, pravite “važeće” aplikacije koristeći gotove elemente radnog okruženja i time se prilagođavate standardima – primer Java API, applet i web pretraživači

Primer 3.5 – elektronsko poslovanje

- Razvoj veb sajta picerije koji omogućava on-line narudžbine
- Cilj nam je da napravimo **radno okruženje koje bi uvelo višekratnu upotrebljivost koda u praksi**
- Treba razviti **funkcionalno radno okruženje** upotrebom **nasleđivanja**, **spajanja, apstraktnih klasa** i **interfejsa**
- Šta ako sutradan dobijemo zahtev za razvoj veb sajta poslastičarnice? Da li ćemo pisati novu aplikaciju? Koliko još porodičnih radnji može koristiti naše okruženje na vebu?
- Ako imamo **dobro i pouzdano radno okruženje**, **programe** bi mogli da nudimo po **povoljnijim cenama**, a pri tom bi oni bili **već isprobani i primenjeni**, što **smanjuje posao oko održavanja i otklanjanja grešaka**

Primer 3.5 – pristup bez ponovne upotrebe koda

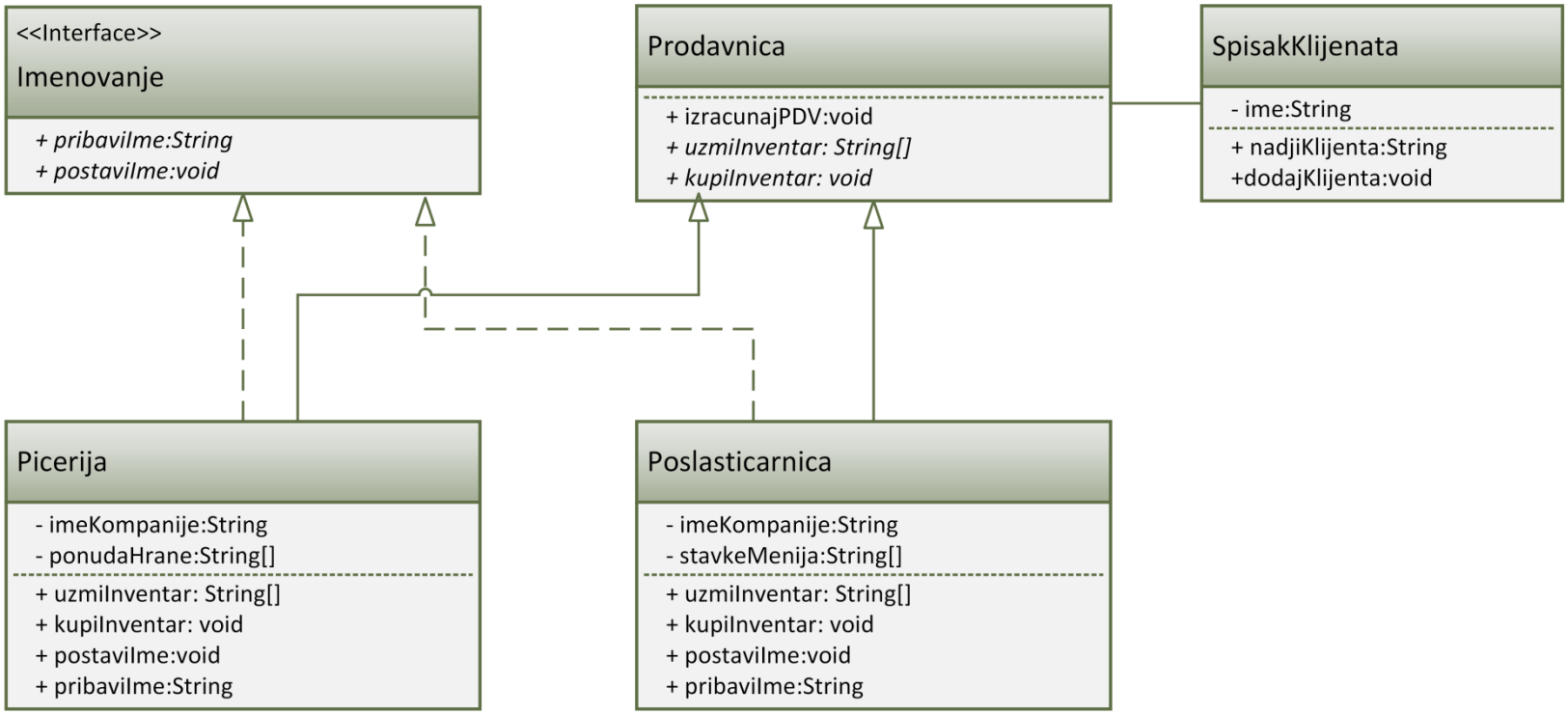


Primer 3.5 – rešenje za elektronsko poslovanje

- Napravićemo **apstraktnu klasu** koja **izdvaja određenu realizaciju** i **interfejs koji modeluje neka ponašanja**
- **Cilj – radno okruženje kojim višekratno koristimo kod**
- Svaka specifična aplikacija biće **vezana samo ugovorom**, nema njenog strogog vezivanja za prilagođene klase
- Sastoji se od:
  - Interfejsa `Imenovanje`, modeluje ponašanja, deo ugovora,
  - Apstraktne klase `Prodavnica`, izdvaja implementaciju, deo ugovora,
  - Klase `SpisakKlijenata`, koju koristimo kroz spajanje i
  - Nove realizacije klase `Prodavnica` za svakog klijenta kroz klase potomke koji je nasleđuju

Primer 3.5 – dijagram klasa

Apstraktni metodi se obeležavaju *italic* slovima



realizacija interfejsa

Primer 3.5 – apstraktna klasa Prodavnica

```
public abstract class Prodavnica {
	private SpisakKlijenata spisakKlijenata;

	public void izracunajPDV() {
		System.out.println("Stopa PDV je 20%!");
	}

	public abstract String[] uzmiInventar();

	public abstract void kupiInventar(String artikal);
}
```

Primer 3.5 – klasa SpisakKlijenata

```
public class SpisakKlijenata {
        private String[] ime;
        private int trenutniBrojKlijenata;
        private int maxBrojKlijenata;

        SpisakKlijenata(){}

        SpisakKlijenata(int maxBrojKlijenata){
                this.maxBrojKlijenata = maxBrojKlijenata;
                this.trenutniBrojKlijenata = 0;
                this.ime = new String[maxBrojKlijenata];
        }

        public String nadjiKlijenta(String ime) {
                for (int i = 0; i < this.trenutniBrojKlijenata; i++) {
                        if (this.ime[i].equals(ime)) {
                                return this.ime[i];
                        }
                }
                return ("Klijent nije pronadjen!");
        }
        ...
```

Primer 3.5 – klasa SpisakKlijenata

```
        ...

        public void dodajKlijenta(String ime) {
                if (this.trenutniBrojKlijenata < this.maxBrojKlijenata) {
                        this.ime[this.trenutniBrojKlijenata++] = ime;
                }
                else{
                        System.out.println("Nema vise mesta u spisku
                                                klijenata!");
                }
        }
}
```

Primer 3.5 – interfejs Imenovanje

```
public interface Imenovanje {

	String pribaviIme();
	void postaviIme(String ime);

}
```

Primer 3.5 – klasa Picerija

```
public class Picerija extends Prodavnica implements Imenovanje {

        private String imeKompanije;

        private String[] ponudaHrane = {
                        "Pica",
                        "Pasta",
                        "Salata",
                        "Kalcona",
                        "Sok",
                        "Pivo"
        };

        public String[] uzmiInventar() {
                return ponudaHrane;
        }

        ...
```

Primer 3.5 – klasa Picerija

```
        ...

        public void kupiInventar(String artikal) {
                System.out.println("\nUpravo ste narucili artikal “
                                        + artikal);
        }

        public String pribaviIme() {
                return imeKompanije;
        }

        public void postaviIme(String ime) {
                imeKompanije = ime;
        }

}
```

Primer 3.5 – klasa Poslasticarnica

```
public class Poslasticarnica extends Prodavnica implements Imenovanje{

	private String imeKompanije;

	private String[] stavkaMenija = {
			"Sladoled",
			"Torta",
			"Krofna",
			"Kafa",
			"Caj",
			"Limunada"
	};

	public String[] uzmiInventar() {
		return stavkaMenija;
	}

	...
```

Primer 3.5 – klasa Poslasticarnica

```
        ...

        public void kupiInventar(String artikal) {
                System.out.println("\nUpravo ste narucili artikal “
                                        + artikal);
        }

        public String pribaviIme() {
                return imeKompanije;
        }

        public void postaviIme(String ime) {
                imeKompanije = ime;
        }

}
```

Primer 3.5 – klasa Main

```
public class Main {
    public static void main(String[] args) {

            Poslasticarnica carigrad = new Poslasticarnica();
            Picerija ciao = new Picerija();

            carigrad.postaviIme("Evropa");
            ciao.postaviIme("Ciao");

            carigrad.kupiInventar("Sladoled");
            ciao.kupiInventar("Pica");
            ...
    }
}
```

Zadatak za rad na času

- Zadatak: dopuniti apstraktnu klasu `Prodavnica`, interfejs `Imenovanje`, klasu `SpisakKlijenata`, kao i izvedene klase `Picerija` i `Poslasticarnica` novim atributima i metodima i proširiti implementacije postojećih metoda, tako da se realizuju funkcionalnosti koje bi se mogle zahtevati u svakodnevnom poslovanju
- Realizovati nove klase `Restoran` i `Knjizara` koje takođe nasleđuju apstraktnu klasu `Prodavnica` i implementiraju interfejs `Imenovanje`
- Klase testirati kreiranjem više objekata u glavnom programu i pozivanjem odgovarajućih izabranih metoda

TAČNOST I ROBUSNOST OO PROGRAMA – GENERISANJE I OBRADA IZUZETAKA