Properties

Dobro je poznato, još sa početka ovog programa, da se klase u Javi generalno sastoje iz dva tipa članova: polja i metoda. Polja su tu da modeluju stanje objekata, dok metode definišu ponašanja. Ukoliko se, prilikom kreiranja klase, poštuju i određena pravila definisanja polja, klasa se može nazvati Java Beanom.

Java Bean

Java Bean je takozvana reusable komponenta, zato što se ponaša kao logička celina, koja jednom napravljena može biti korišćena u različitim situacijama. Pravila koja je potrebno poštovati da bi se klasa nazvala Beanom odnose se na definisanje polja. Polja moraju biti privatna, dok se za njihovu manipulaciju definišu javne get i set metode. Sledeći primer ilustruje jedan Java Bean:

```
public class Person {
      private String name;
      public String getName() {
      return name;
      }
public void setName(String name) {
      this.name = name;
   }
}
```

Po konvenciji Java Beans, nazivi privatnih polja pišu se malim slovima, dok se nazivi get i set metoda formiraju dodavanjem prefiks reči get ili set na naziv svojstva koja se u tom slučaju piše početnim velikim slovom.

Sve ovo je već više puta pominjano u dosadašnjem toku programa.

Java FX properties

Nakon kraćeg uvoda, čiji je cilj da vas podseti na bitne aspekte Java jezika koji će vam umnogome pomoći u razumevanju onoga što sledi, biće objašnjen pojam propertija u Java FX-u.

Naime, Java FX specifični propertiji predstavljaju veliki iskorak u odnosu na tradicionalne Java Bean propertije. Java FX propertiji su takozvani *observable* propertiji, što znači da sve zainteresovane strane mogu veoma lako da prate njihovu vrednost. Oni su takođe i objekti, bez obzira na to da li reprezentuju neki prost tip ili ne reprezentuju. Tako za proste tipove int i double postoje Integer Property i Double Property. Svaki properti definiše se apstraktnom klasom, a za krajnju upotrebu koriste se neke od klasa koje nasleđuju takve apstraktne klase. Uglavnom za svaki tip postoje po dve implementacije, jedna za propety koji dozvoljava samo čitanje, a druga za property koji dozvoljava i čitanje i pisanje.

Tako se na primer jedan property koji može imati celobrojnu vrednost predstavlja na sledeći način:

```
IntegerProperty counter = new SimpleIntegerProperty(100);
```

Integer Property je apstraktna klasa, dok je Simple Integer Property implementacija. Broj 100 označava početnu vrednost.

U tabeli je dat prikaz Java FX propertija i vrednosti za koje se oni koriste:

tip	read/write property	read-only property	read/write implementacija
int	IntegerProperty	ReadOnlyIntegerProperty	SimpleIntegerProperty
long	LongProperty	ReadOnlyLongProperty	SimpleLongProperty
float	FloatProperty	ReadOnlyFloatProperty	SimpleFloatProperty
double	DoubleProperty	ReadOnlyDoubleProperty	SimpleDoubleProperty
boolean	BooleanProperty	ReadOnlyBooleanProperty	SimpleBooleanProperty
String	StringProperty	ReadOnlyStringProperty	SimpleStringProperty
Object	ObjectProperty	ReadOnlyObjectProperty	SimpleObjectProperty

Tabela 3.1

U prethodnom primeru u kome je definisan counter Integer Property, pri kreiranju propertija prosleđena je samo početna vrednost. Java FX propertiji mogu sadržati tri informacije:

- referencu na roditeljski bean,
- naziv i
- inicijalnu vrednost.

Konkretne implementacije apstraktnih klasa definišu preklopljene konstruktore, tako da je moguće prilikom kreiranja specifikovati sve, nijednu ili samo neku od prikazanih vrednosti. Ti konstruktori izgledaju ovako:

```
SimpleIntegerProperty()
SimpleIntegerProperty(int initialValue)
SimpleIntegerProperty(Object bean, String name)
SimpleIntegerProperty(Object bean, String name, int initialValue)
```

Podrazumevana vrednost propertija zavisi od njihovog tipa. Ona je nula za proste numeričke tipove, false za logičke, a null za objektne tipove.

Property može biti samostalan ili deo nekog beana. Parametar bean se odnosi upravo na referencu beana koji sadrži property. Ukoliko se ne navede, ima vrednost null.

Parametar name se odnosi na naziv propertija. Ukoliko se navede, vrednost je prazan string.

Za dobavljanje ovih vrednosti svaka klasa propertija poseduje metode get Bean i get Name koje vraćaju referencu na bean i naziv propertija, respektivno.

Rukovanje Java FX propertijima

Java FX propertiji poseduju dva para metoda za manipulaciju vrednostima. To su parovi get/set i getValue/setValue.

Metode get i set imaju specijalnu upotrebu samo kod primitivnih tipova, kao što je na primer int. Tako bi, na primeru nešto ranije prikazanog propertija Integer Property, metoda get vratila vrednost tipa int, dok bi metoda set za postavljanje vrednosti prihvatala takođe prost tip int. Na istom primeru propertija Integer Property, metode get Value i set Value baratale bi objektnim Integer tipom. Metoda get Value vratila bi objektnu vrednost tipa Integer, dok bi metoda set Value kao tip parametra očekivala vrednost tipa Integer. Kod tipova koji su striktno objektni, kao što je to slučaj sa tipom String, oba para metoda imaju istu ulogu.

Sledeći blok koda ilustruje postavljanje i čitanje vrednosti jednog Java FX propertija.

```
IntegerProperty counter = new SimpleIntegerProperty(1);
int counterValue = counter.get();
System.out.println("Counter:" + counterValue);
counter.set(2);
counterValue = counter.get();
System.out.println("Counter:" + counterValue);
```

Prikazani kod ima kao rezultat sledeći ispis u konzoli:

```
Counter:1
Counter:2
```

Na početku ove lekcije dat je primer klase Person definisane po Java Bean konvenciji. Ista klasa kreirana korišćenjem Java FX propertija bi izgledala ovako (klasi su radi potpunijeg primera dodata polja i konstruktori):

```
public class Person {

private final StringProperty firstName = new
SimpleStringProperty(this, "firstName", "");
private final StringProperty lastName = new SimpleStringProperty(this,
"lastName", "");
private final IntegerProperty age = new SimpleIntegerProperty(this,
"age", 0);

    public Person() {

    }

    public Person(String firstName, String lastName, int age) {
        this.firstName.set(firstName);
        this.lastName.set(lastName);
        this.age.set(age);

    }

    public Person(String firstName) {
        this.firstName.set(firstName);
    }
```

```
    public Person(String firstName, String lastName) {
        this.firstName.set(firstName);
        this.lastName.set(lastName);

    }

    public String getFirstName() {
        return firstName.get();
    }

    public void setFirstName(String firstName) {
        this.firstName.set(firstName);
    }

    public StringProperty firstNameProperty() {
        return firstName;
    }

    public String getLastName() {
        return lastName.get();
    }

    public void setLastName(String lastName) {
        this.lastName.set(lastName);
    }

    public StringProperty lastNameProperty() {
        return lastName;
    }

    public int getAge() {
        return age.get();
    }

    public void setAge(int age) {
        this.age.set(age);
    }

    public IntegerProperty ageProperty() {
        return age;
    }

}
```

S obzirom na to da Java FX propertiji automatski sadrže get i set metode, kao što se moglo videti u prethodnom primeru upotrebe ovih propertija, navođenje posebnih get i set metoda nije obavezno. Ipak, zbog kompatibilnosti sa starijom specifikacijom, česta je praksa da se ove dve metode simuliraju, na način prikazan pri definisanju Person klase. Za svaki property u klasi Person postoje tri pomoćne metode. Jedna, koja isporučuje Java FX property, i preostale dve koje isporučuju ili postavljaju vrednost u stilu Java Bean konvencije.

Lazily instanciranje

Do sada ste mogli da naslutite moć koja se krije iza Java FX propertija. Ali sa moćnim funkcionalnostima dolazi i nekoliko slabosti. S obzirom na to da su svi Java FX propertiji objektni tipovi, postojanje velikog broja ovakvih propertija u okviru jednog beana može znatno da naruši performanse aplikacije. U takvim situacijama se pribegava korišćenju odloženog instanciranja propertija.

Podrazumevano, kao što je slučaj u primeru klase Person, koji je nešto ranije prikazan, svi Java FX propertiji instanciraju se onda kada se instancira i Person objekat. Takvo instanciranje propertija može se nazvati *eager* instanciranje.

Za odloženo učitavanje propertija ili lazy učitavanje, potrebno je klasi dodati logiku koja će obavaljati takav zadatak. Pre samog primera, potrebno je još napomenuti da je odloženo instanciranje dobro koristiti samo u nekim situacijama, i to kada znamo da će property u najvećem broju slučajeva imati podrazumevanu vrednost, ili kada znamo da ne postoji potreba za specijalnim osobinama Java FX propertija, kao što su praćenje i binding (više o ovom pojmu u nastavku).

Zapravo, odloženo instanciranje Java FX propertija ne povlači za sobom upotrebu bilo kakvih specijalnih funkcionalnosti Java FX biblioteka, već se sve zasniva na dobro poznatim osnovnim osobinama jezika.

Jedino pitanje koje se može javiti kod ovakvog scenarija jeste kada će doći do instanciranja punog Java FX propetija. U zavisnosti od toga, postoje dva načina na koje je moguće realizovati odloženo instanciranje.

Half lazy

```
package main;

import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;

public class Person {

    private DoubleProperty weight;
    private double _weight = 150;

    public double getWeight() {
        return (weight == null) ? _weight : weight.get();
    }

    public void setWeight(double newWeight) {

        if (weight != null || !(_weight == newWeight)) {
            weightProperty().set(newWeight);
        }

    }

    public DoubleProperty weightProperty() {
        if (weight == null) {
        weight = new SimpleDoubleProperty(this, "weight", _weight);
        }
        return weight;
    }

}
```

Person klasa sadrži polje za čuvanje težine osobe. Interesantno je da je u klasi pored Java FX propertija definisano i jedno obično Java polje. Takođe, može se uočiti da je Java FX property neinstanciran inicijalno.

Pored dve set i get metode za rukovanje poljem, ova klasa poseduje još jednu metodu. Logika je sledeća. Težina je inicijalno postavljena na 150. Metodom get Weight se može pročitati vrednost polja. Ukoliko Java FX polje nije instancirano, kao vrednost će biti vraćena vrednost polja _weight. Ukoliko je Java FX property instanciran, biće vraćena njegova vrednost.

Metoda set Weight se koristi za postavljanje vrednosti polja. Ovo je trenutak u kome će doći do instanciranja Java FX propertija. Jednostavno, ukoliko je nova vrednost koja se prosleđuje različita od vrednosti _weight polja, doći će do instanciranja Java FX propertija. U situacijama kada Java FX property već postoji samo će biti promenjena njegova vrednost.

Na kraju dolazimo i do metode koja obavlja instanciranje Java FX polja i njega isporučuje kao svoju povratnu vrednost. Ukoliko je Java FX property null, biće obavljeno instanciranje i kao inicijalna vrednost postavljena trenutna vrednost _weight polja. U situacijama kada je Java FX property već instanciran ova metoda vratiće tu instancu.

Pravilna upotreba ovako definisane klase bi izgledala:

```
Person p = new Person();
double weight = p.getWeight();
```

Na ovaj način, prilikom iščitavanja vrednosti, neće doći do instanciranja Java FX propertija. Ipak, prilikom postavljanja vrednosti polja, uvek dolazi do instanciranja Java FX propertija, a takvo ponašanje nije uvek poželjno.

Full lazy

Zbog toga postoji i drugi pristup odloženom učitavanju propertija, gde će Java FX property biti učitan samo onda kada se eksplicitno zahteva. Takav scenario zahteva samo blagu modifikaciju postojećeg primera.

```
package main;

import javafx.beans.property.DoubleProperty;
import javafx.beans.property.SimpleDoubleProperty;

public class Item {

    private DoubleProperty weight;
    private double _weight = 150;

    public double getWeight() {
        return (weight == null) ? _weight : weight.get();
    }

    public void setWeight(double newWeight) {

        if (weight == null) {
            _weight = newWeight;
        } else {
            weight.set(newWeight);
        }
    }

    public DoubleProperty weightProperty() {
        if (weight == null) {
```

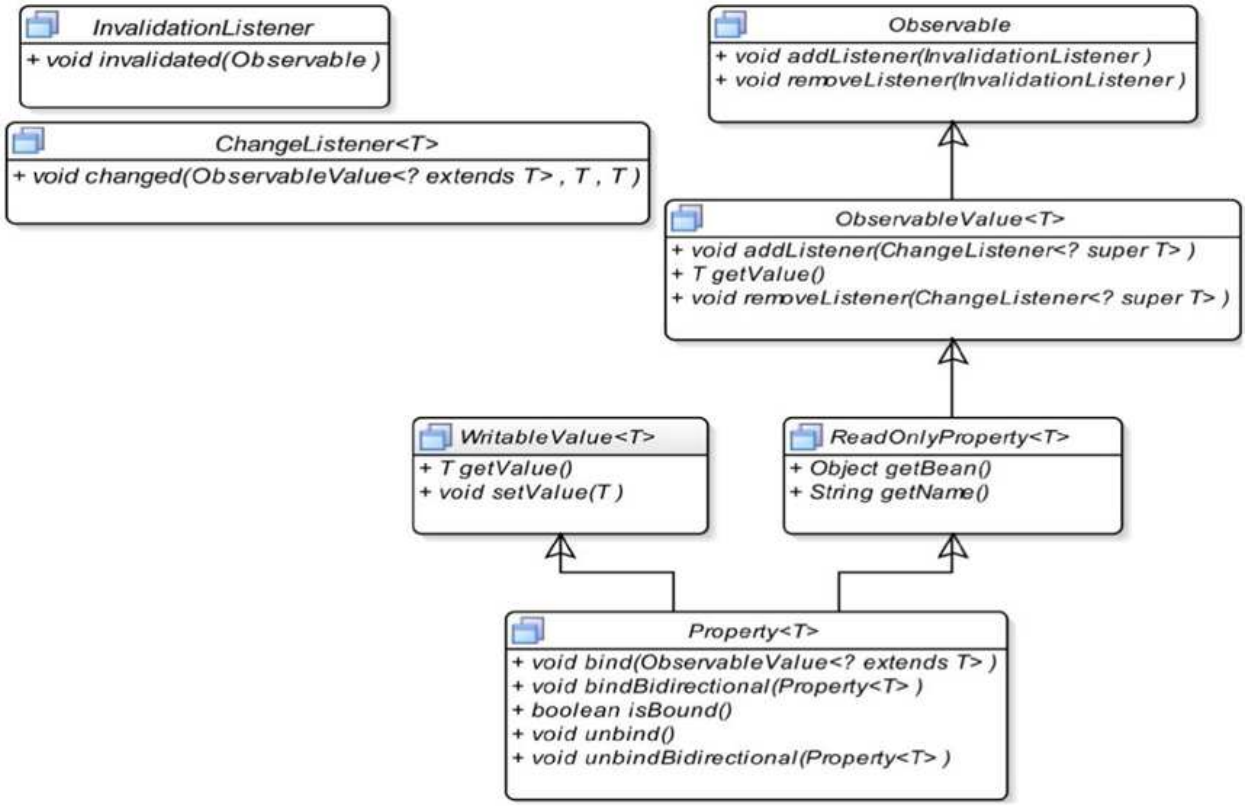
```
            weight = new SimpleDoubleProperty(this, "weight", _weight);
            }
            return weight;
        }

    }
```

Ono što je promenjeno jeste metoda za postavljanje vrednosti polja set Weight. Sada se u okviru ove metode proverava da li je Java FX property instanciran. Ukoliko nije, prosleđena nova vrednost se postavlja za vrednost običnog Java polja. Samo u situacijama kada je Java FX property već instanciran vrednost će biti postavljena unutar Java FX polja.

Ovo znači da do instanciranja Java FX propertija neće doći sve dok se to eksplicitno ne zatraži i to metodom weight Property.

Java FX property klasna hijerarhija

Sledeći klasni dijagram pomoći će vam da bolje razumete logiku iza Java FX propertija.



3.1 – Java FX Property klasni dijagram

Osnovu ove hijerarhije čini interfejs Observable. Njegova glavna funkcionalnost je mogućnost generisanja invalidation događaja kada se status sadržaja promeni iz validnog u nevalidan ili obrnuto.

Sledeći bitan interfejs, jednu lestvicu niže u hijerarhiji, jeste Observable Value interfejs, koji nasleđuje interfejs Observable. Njegova glavna osobina je rukovanje vrednošću, tako da on omogućava generisanje događaja koji simbolizuju promenu vrednosti.

Upravo opisani događaji su veoma bitni pri radu sa Java FX propertijima.

Invalidation Events

Java FX property generiše *invalidation* događaj onda kada se status njegove vrednosti promeni sa validnog na nevalidan ili obrnuto, prvi put. Ovo znači da kada nevalidan property postane ponovo nevalidan, ovaj događaj se neće okinuti.

Ovo je događaj na koji se mogu pretplatiti instance tipa Observable, a to je zapravo tip koji ne rukuje direktno vrednošću polja. Stvarna vrednost propertija biće rekalkulisana tek onda kada bude i zatražena, tako da se sa sigurnošću ne može znati da li se vrednost uistinu i promenila.

```
IntegerProperty counter = new SimpleIntegerProperty(100);
counter.addListener(InvalidationTest::invalidated);
```

Evo jednog Java FX propertija i registrovanja slušača Invalidation događaja.

U kodu se koristi lambda izraz i referenca na metodu, što su funkcionalnosti Java 8 programskog jezika, tako da je ekvivalentan kod korišćenjem anonimnih klasa sledeći:

```
counter.addListener(new InvalidationListener() {
@Override
public void invalidated(Observable prop) {
InvalidationTest.invalidated(prop);
}});
```

Change Events

Java FX property emituje Change događaj svaki put kada se vrednost propertija promeni. U tim situacijama se poziva changed() metoda Change Listenera kojoj se prosleđuju tri vrednosti: referenca na objekat propertija, stara vrednost i nova vrednost.

Prilikom prosleđivanja ovog događaja, s obzirom na to da je neophodno isporučiti novu vrednost, nju je prethodno potrebno proračunati. Ovo znači da je korišćenjem change događaja nemoguće vrednost proračunavati odloženo (lazy loading).

Sledi demonstracija dodavanja slušača Change događaja.

```
IntegerProperty counter = new SimpleIntegerProperty(100);

counter.addListener(new ChangeListener<Number>() {
@Override
public void changed(ObservableValue<? extends Number> prop,
Number oldValue,
Number newValue) {
```

```
System.out.print("Counter changed: ");
System.out.println("Old = " + oldValue + ", new = " + newValue);
}});
```

Pitanje

Koji je osnovni interfejs koji implementira Java FX property?

a) Observable
b) Observable Value
c) Writable Value
d) Property

Observable je osnovni interfejs koji implementiraju Java FX propertiji. Njegova glavna funkcionalnost je mogućnost generisanja invalidation događaja kada se status sadržaja promeni iz validnog u nevalidan ili obrnuto.

Rezime

- Klase u Javi sastoje se iz dva tipa članova: polja i metoda; polja modeluju stanje objekata, dok metode definišu ponašanja.
- Java Bean je takozvana reusable komponenta, kreirana uz poštovanje Java Bean specifikacije, i ponaša se kao logička celina koja, jednom napravljena, može biti korišćena u različitim situacijama.
- Java FX definiše specijalne Java FX propertije sa nizom moćnih funkcionalnosti.
- Java FX propertiji su objekti, bez obzira na to da li reprezentuju neki prost tip ili ne reprezentuju.
- Java FX propertiji poseduju dva para metoda za manipulaciju vrednostima. To su parovi get/set i getValue/setValue.
- S obzirom na to da su svi Java FX propertiji objektni tipovi, postojanje velikog broja ovakvih propertija u okviru jednog beana može znatno da naruši performanse aplikacije.
- Poboljšanje performansi aplikacije sa Java FX propertijima postiže se odloženim instanciranjem propertija.
- Java FX propertiji implementiraju interfejs Observable; njegova glavna funkcionalnost je mogućnost generisanja invalidation događaja kada se status sadržaja promeni iz validnog u nevalidan ili obrnuto.
- Java FX propertiji implementiraju interfejs Observable Value; glavna osobina Observable Value interfejsa je rukovanje vrednošću, tako da on omogućava generisanje događaja koji simbolizuju promenu vrednosti propertija.