2.1. Manipulacija Java projektima u Eclipse - u

2.2. Debug perspektiva u Eclipsu

2.3. Kretanje kroz program u Debug mode-u

2.4. Pregled stanja objekata u Debug mode-u

2.5. Argumenti main metode

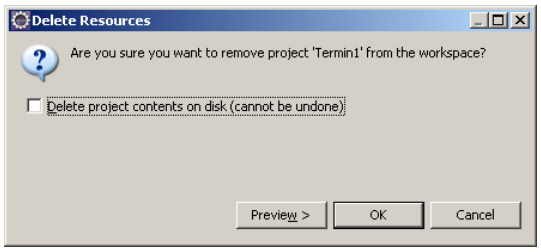
2.6. Zadaci

2.1. Manipulacija Java projektima u Eclipse - u

Pored kreiranja novog Java projekta, postoji još nekoliko operacija sa projektima iz Eclipse okruženje koje su neophodne za rad:

a) Brisanje Java projekta iz Workspace-a (bez brisanja sadržaja na disku)

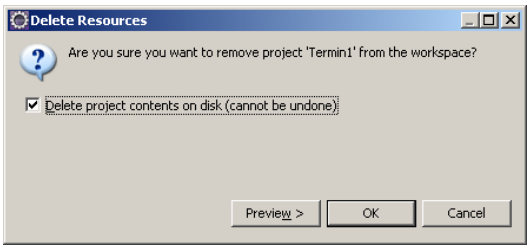
Brisanje projekta iz Workspace -a se obavlja na sledeći način: selektuje se projekat u Package Exploreru koji se želi obrisati -> Desni klik -> Delete



Ukoliko se ne izabere opcija „Delete project contents on disk“ projekat se briše samo iz logičkog Workspace-a Eclipsa dok kompletan folder sa projektom ostaje nedirnut na lokaciji na kojoj se nalazi.

b) Brisanje Java projekta iz Workspace-a (brisanje i sadržaja na disku)

U slučaju da se prilikom brisanja projekta iz Workspace-a izabere opcija „Delete project contents on disk“, projekat će biti i logički i fizički izbrisan iz Workspace-a bez mogućnosti Undo akcija. Zbog toga treba biti veoma oprezan sa ovom akcijom.



Pokrenuti Eclipse i fizički obrisati projekat „Termin1“ iz Workspace-a. Nakon brisanja proveriti kroz Windows Exporer da li je projekat obrisan i fizički iz Workspace foldera.

c) Import postojećeg Java projekta u Workspace

Pored kreiranja novog Java projekta postoji još jedan način otvaranja Java projekta u Eclipse okruženju. Ovo omogućava Import opcija koja kompletan Java projekat „uvlači“ u Workspace. Da bi se projekat uvezao u Eclipse Workspace potrebno je: File -> Import -> General -> Existing Project into Workspace i odabrati putanju do foldera sa Eclipse projektom.

Izvršiti import projekta Termin1 u Eclipse Workspace.

2.2. Debug perspektiva u Eclipsu

Debuger predstavlja alat u razvojnom programskom okruženju koji pruža mogućnost:

a) zaustavljanja izvršavanja aplikacije u bilo kojoj liniji koda
b) pristupanje sadržaju i stanju objekata unutar programa u bilo kom trenutku.

Debuger se koristi za:

a) otklanjanje grešaka u aplikaciji (pronalaženje gde i zašto program ne radi)
b) upoznavanje novih projekata
c) bolje razumevanje načina rada programskog jezika Java i objektno orijentisanog programiranja

Da bi se koristio debuger u Eclipse okruženju potrebno je naučiti:

a) kako se kretati kroz program u debug režimu rada i
b) kako videti stanje željenog objekta u programu

Pre samog ulaska u debug mod, da bi mogli precizno odrediti ne samo klasu već i redni broj reda na kojoj se nalazi određena linija koda potrebno je da uključimo prikaz rednog broja linije unutar klase: Window -> Preferences -> General -> Editors -> Text Editors i izabrati opciju „Show line numbers“.

Importovati projekat „Termin2“ i pogledati izmene koje su napravljene u klasi MyLibrary. Sada je unutar main metode dodato i iznajmljivanje slučajno izabranih 5 knjiga slučajno izabranim članovima biblioteke. Metoda checkOut proverava da li je željena knjiga slobodna i ukoliko jeste ona se izdaje članu biblioteke.

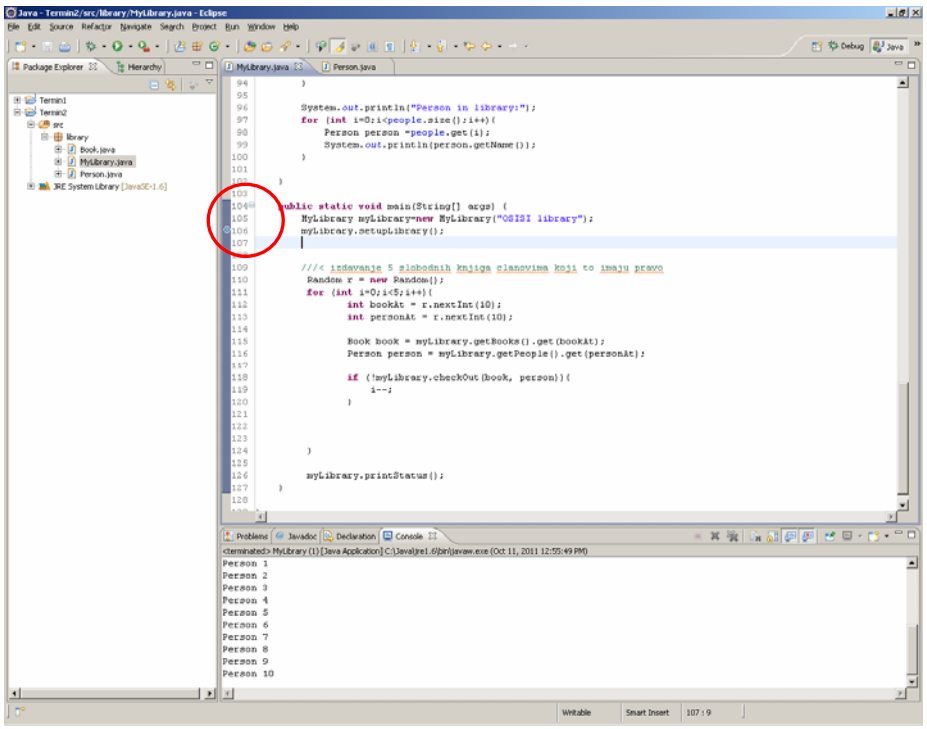
Pitanje: Šta predstavlja toString metoda klase Person i da li se ova metoda implicitno poziva u programu?

Da bi započeli debug sesiju u našem programu potrebno je da:

a) odredimo mesto u našem programu u kome će se zaustaviti izvršavanje programa
b) pokrenemo aplikaciju u debug modu.

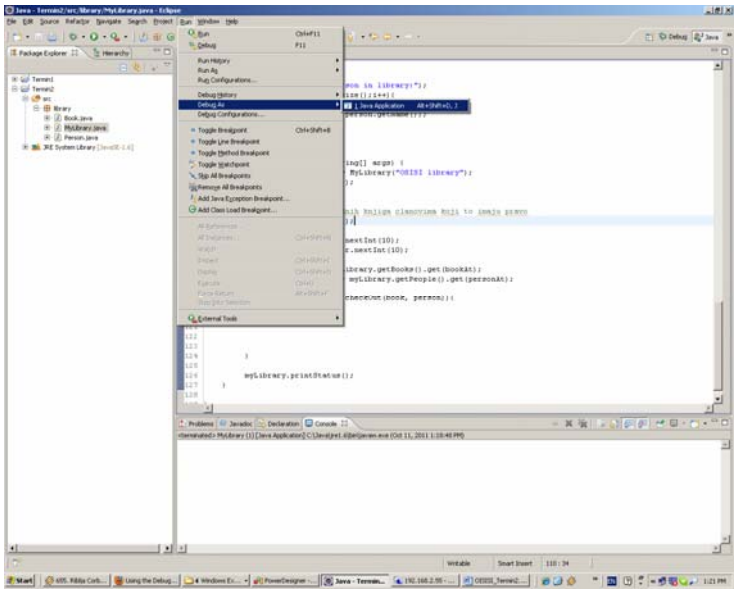
Otvorićemo klasu MyLibrary i postavićemo mesto u kome će se zaustaviti izvršavanje programa. Ovo mesto se zove prekidna tačka izvršavanja (break point) i koristi se samo u debug režimu rada. U programu moguće je imati više od jedne prekidne tačke. Rad u debug modu nije moguć bez barem jedne prekidne tačke.

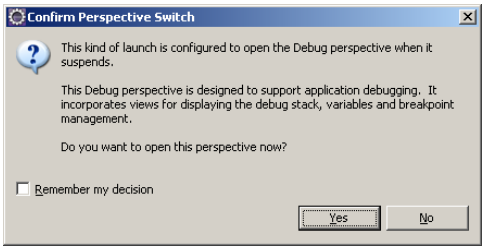
Prekidna tačka se postavlja tako što se u klasi projekta koja je otvorena u editor delu Eclipse okruženja mišem izvrši dupli klik na levu ivicu editora, neposredno pre broja linije koda.



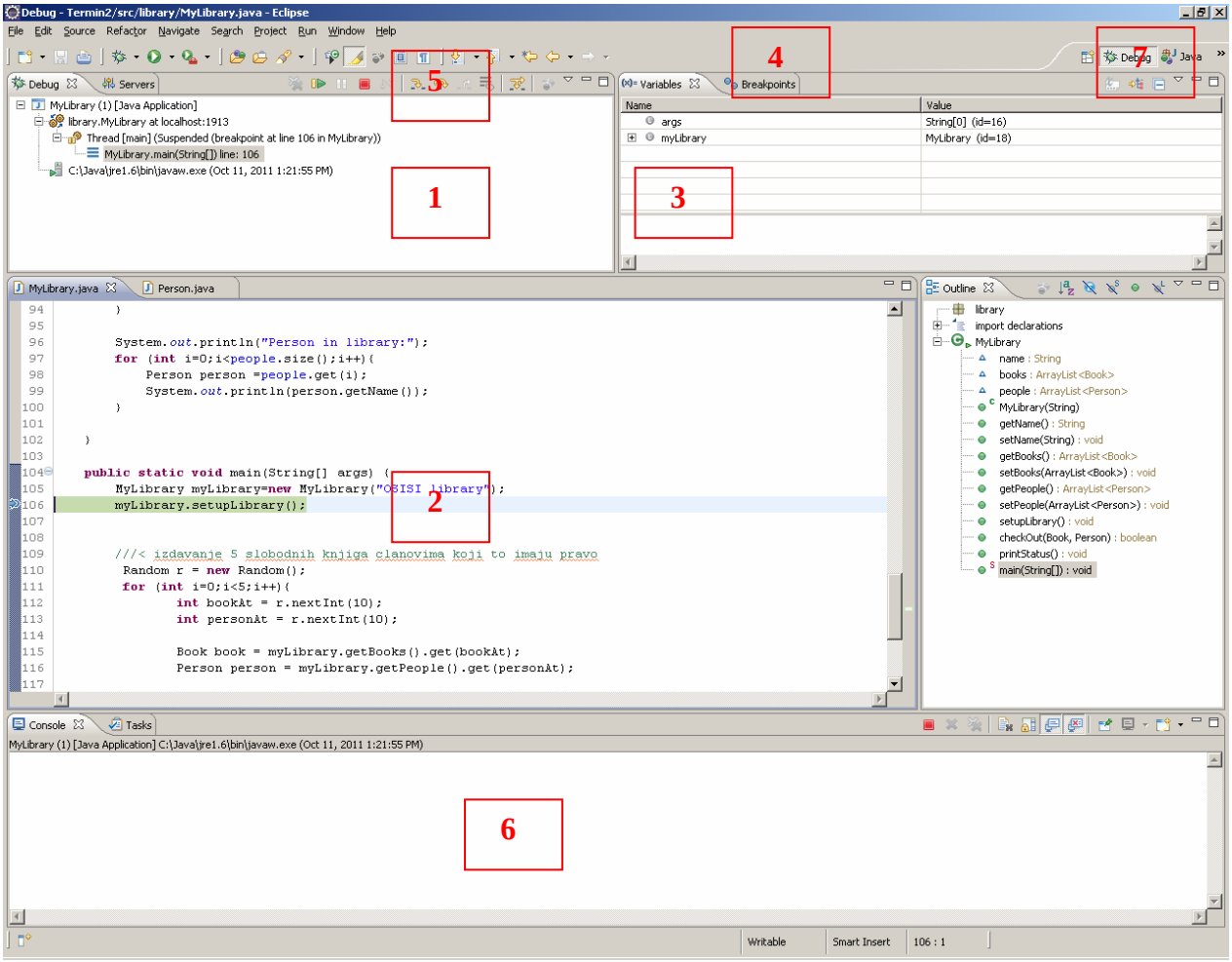
Akcija postavljanja prekidne tačke vizuelno biće obeležena plavim kružićem na liniji koda u kojoj je ona postavljena (u primeru sa slike 106. linija koda klase MyLibrary). Prekidna tačka govori debuger-u gde da zaustavi izvršavanje programa. Prekidna tačka se briše identičnom akcijom kao i prilikom postavljanja (dupli klik nad istim mestom).

Ukoliko se aplikacija pokrene u normalnom (Java) režimu rada sa komandom Run As Java Application, prekidne tačke nemaju apsolutno nikakav uticaj na rad aplikacije. Međutim ukoliko se aplikacija pokrene u debug modu aplikacija će neposredno pre izvršavanja linije koda u kojoj je postavljena prekidna tačka zaustaviti svoje izvršavanje. Aplikacija se pokreće u debug modu na sledeći način: **Run -> Debug As -> Java application.** Neposredno pre pokretanja aplikacije tražiće se potvrda promene aktivne perspektive Eclipse okruženja iz Java u Debug perspective. Rad u debug modu zahteva naravno i promenu perspektive u Debug:





Debug perspektiva prilagođava izgled Eclipse okruženja radu u debug modu:



1. Debug View - prozor koji prikazuje naziv i poziciju trenutno aktivne programske niti

2. Java Editor - prikazuje liniju koda koja će se naredna izvršiti. Ova linija koda je posebno označana (u primeru linija koda je selektovana zelenom pozadinom)

3. Variable view - omogućava uvid u vrednosti objekata aplikacije

4. Breakpoints view - prikaz svih aktivnih prekidnih tačaka

5. Toolbar debug mode-a - akcije koje omogućavaju kretanje kroz debug mode.

6. Sistemska konzola Eclipse-a.

7. Pregled mogućih perspektiva Eclipse okruženja.

2.3. Kretanje kroz program u Debug mode-u

Toolbar debug mode-a predstavlja najvažniji deo Debug perspektive koji omogućava kretanje kroz program:

1. Resume akcija - omogućava nastavak izvršavanja programa ili do sledeće prekidne tačke ili do izlazne tačke aplikacije

2. Terminate akcija - zaustavljanje debug sesije, odnosno završetak rada aplikacije

3. Step over akcija - izvršava trenutno selektovanu liniju koda i staje na narednoj liniji.

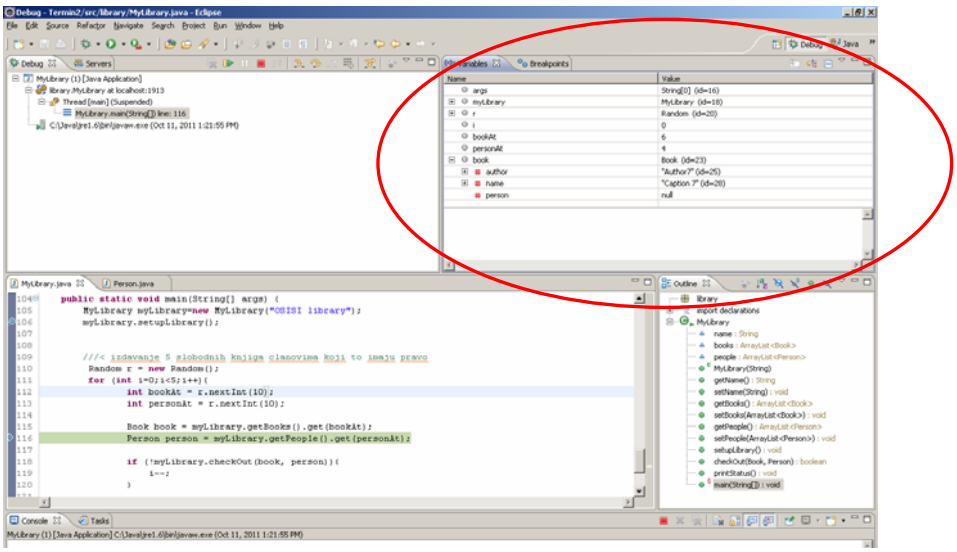
4. Step into akcija - ulazi unutar metode ili konstruktura koji je trenutno selektovan i zaustavlja izvršavanje aplikacije na vrhu metode ili konstruktora

5. Step return metoda - završava izvršavanje trenutno aktivne metodueu kojoj se program nalazi i vraća kontrolu na metodu pozivaoca.

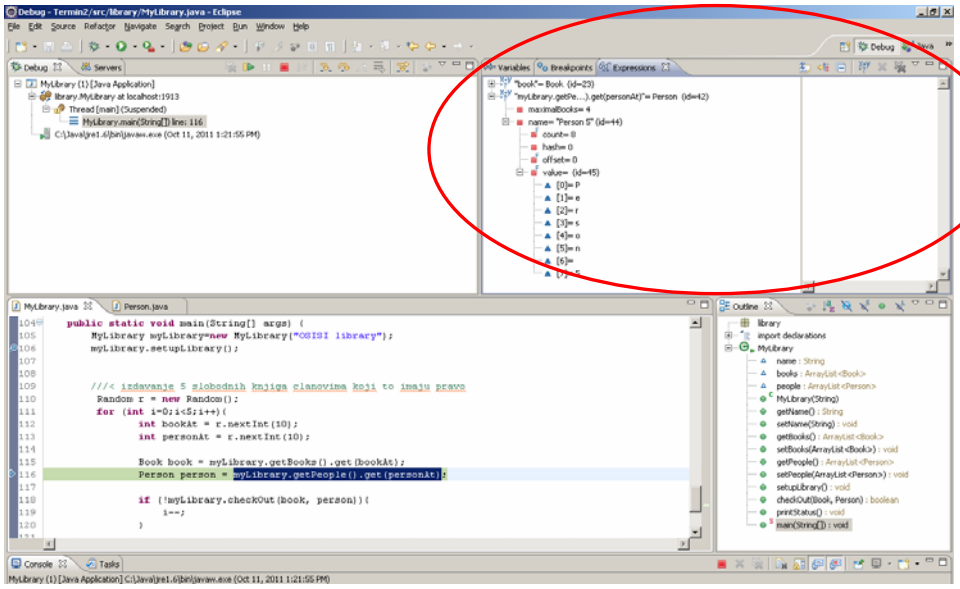
Izvršiti prolazak kroz kompletan program u Termin2 projektu u debug režimu rada.

2.4. Pregled stanja objekata u Debug mode- u

Sadržaj i vrednost svih objekata koji su u datom trenutku aktivni u debug režimu rada moguće je videti u Variable view prozoru debug perspektive:



U slučaju da korisnik želi da dobije vrednost izvršavanja neke metode postoji mogućnost akcije **Watch**. Selekcijom određenog dela koda trenutno aktivne linije koda u debug modu moguće je dobiti vrednost selektovanog dela koda i pre njegovog izvršavanja. Na primeru sa slike selektovan je deo koda: myLibrary.getPeople().get(personAt) -> Desni klik nad selektovanim kodom -> Watch. U Expression prozoru se pojavljuje vrednost selektovanog iskaza.

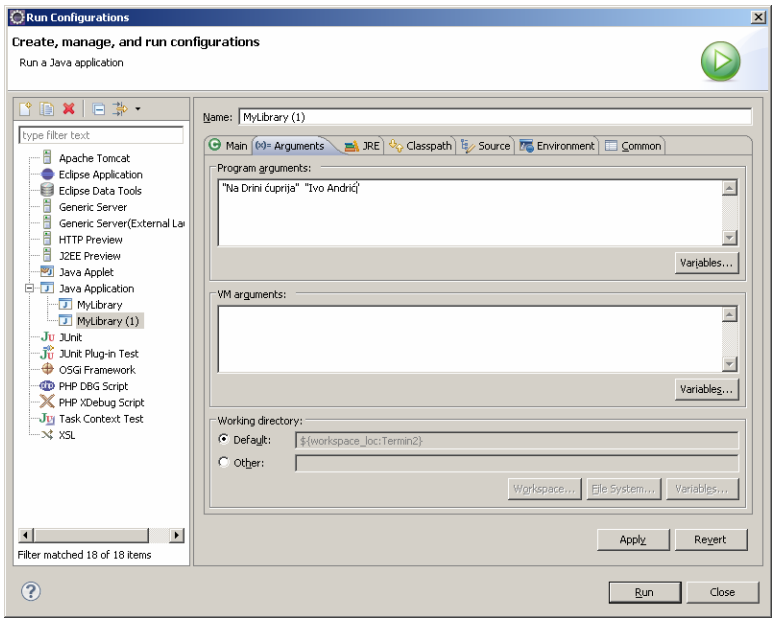


Ista akcija je moguća i za bilo koju drugu metodu i/ili objekat čije nas stanje (vrednosti atributa) interesuju.

2.5. Argumenti main metode

Argumenti main metode pružaju mogućnost komunikacije između Java aplikacije i pozivaoca aplikacije, odnosno obezbeđuju mehanizam prenošenja parametara iz spoljnog sveta u aplikaciju prilikom njenog pokretanja.

Argumenti u main metodu ulaze kao niz String objekata. Pored pokretanja Java aplikacije i prosleđivanja vrednosti parametara iz komandne linije (java Sort friends.txt), i samo Eclipse okruženje pruža mogućnost prosleđivanja vrednosti argumenata u main metodu aplikacije: Run -> Run configurations -> kartica Arguments



Kartica Arguments služi za unos vrednosti ulaznih argumenata aplikacije. Svi argumenti se definišu između navodnika i radvojeni su praznim mestom.

Način preuzimanja vrednosti parametara unutar main metode prikazan je u primeru

```
public static void main(String[] args) {
      MyLibrary myLibrary=new MyLibrary("OSISI library");
      myLibrary.setupLibrary();

      if (args.length==2){
            String name = args[0];
            String author = args[1];
            Book book = new Book(name, author, null);

            myLibrary.getBooks().add(book);
      }

       myLibrary.printStatus();
}
```

2.6. Zadaci

1. Izmeniti projekat Termin2 tako što se broj knjiga koje svaki korisnik može u jednom trenutku iznajmiti (vrednost atributa maximalBooks klase Person) prosleđuje kao argument main metode i koristi se prilikom instanciranja svakog objekta klase Person.

2. Pogledati video tutoriale za rad u debug mode-u:
http://sourceforge.net/projects/eclipsetutorial/files/3.%20Debugger%20Tutorial/