PT&SCHE

Introduction of part-time and short cycle studies in Serbia

Funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication [communication] reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

UNIVERZITET
METROPOLITAN
BEOGRAD

KI301 - KONSTRUISANJE SOFTVERA

Konstrukcioni alati

Lekcija 06

PRIRUČNIK ZA STUDENTE

KI301 - KONSTRUISANJE SOFTVERA

Lekcija 06

KONSTRUKCIONI ALATI

- Konstrukcioni alati
- Poglavlje 1: Konstrukcioni alati - uvod
- Poglavlje 2: Vežba - Editori za kodiranje
- Poglavlje 3: CASE alati
- Poglavlje 4: UML alati
- Poglavlje 5: Alati za testiranje i za debugging
- Poglavlje 6: Vežba - Alat FindBugs
- Zaključak

L6: KONSTRUKCIONI ALATI

L6: Konstrukcioni alati - Sadržaj

Sadržaj:

- Konstrukcioni alati - uvod
- Vežba – Editori za kodiranje
- CASE alati
- UML alati
- Alati za testiranje i debugging
- Vežba - Java alati za testiranje
- Vežba- Alat FindBugs

Poglavlje 1

Konstrukcioni alati - uvod

KONSTRUKCIONI ALATI

Neki programeri rade godinama pre nego što otkriju neke vrlo korisne alate.

Programski alati mogu da dramatično smanje vreme konstruisanja softvera. Naime , ako koristite najnovije alate, i ako ste familijarni sa tim alatima, produktivnost može da se poveća za npr. 50%. Ovi alati takodje smanjuju onaj dosadni deo posla koji je prisutan kod programiranja, npr. neke stvari koje se ponavljaju u programima. Dobri programerski alati su vrlo korisni za programere. Ovde posmatramo samo konstrukcione alate, dok se ovde ne bavimo alatima za specifikaciju zahteva i alatima za upravljanje projektima. Neki programeri rade godinama pre nego što otkriju neke vrlo korisne alate. Cilj ovog poglavlja je da se izvrši pregled raspoloživih alata, što pomaže programerima da otkriju da li su prevideli neki alat.

Evo liste nekih najvažnijih alata:

- Dizajnerski alati ,
- Editori izvornog koda
- Alati za upravljanje verzijama (version.control tools) koji omogućuju rad sa namnoženim verzijama nekog programa
- Kompajleri i linkeri
- Alati za izgradnju softvera (build tools)
- Biblioteke programa (code libraries)
- Debugging alati ,
- Alati za testiranje
- Prevodioci koda (prevode program iz jednog jezika u drugi)
- Analizatori kvaliteta programa (ovi alati vrše tzv. statičku analizu programa, npr. syntax checkers, semantics checkers, metrics reporters)
- Alati za refaktorisanje (refactorers,)
- Rečnici podataka (data dictionaries): rečnik podataka u nekom projektu to je baza podataka koja sadrži opise svih značajnih podataka u projektu
- alati za podešavanje programa (code-tuning alati)

DIZAJNERSKI ALATI

S jedne strane, grafički dizajnerski alati su samo divni paketi za crtanje. Ali ovi alati nude više u odnosu na jednostavne tj. obične alate za crtanje.

Postojeći dizajnerski alati su uglavnom grafički alati za kreiranje dizajnerskih dijagrama. Dizajnerski alati su ponekad ugradjeni u CASE (computer-aided software engineering) alat koji ima šire funkcije od dizajnerskih funkcija, dok neki prodavci (vendors) prodaju samostalne (standalone) dizajnerskealate kao CASE alate. Grafički dizajnerski alati omogućuju da se dizajn predstavi u uobičajenim grafičkim notacijama dizajna:

UML, Arhitektonski blok dijagrami ,Hijerarhijske karte , Entitetski dijagrami , Klasni dijagrami

Neki grafički dizajnerski alati podržavaju samo jednu od ovih grafičkih notacija, a neki podržavaju čitav niz ovih grafičkih notacija.

S jedne strane, ovi grafički dizajnerski alati su samo divni paketi za crtanje. Ali ovi alati nude više u odnosu na jednostavne tj. obične alate za crtanje.

- Npr. ako nacrtate neki dizajnerski dijagram i sklonite ili dodate jedan element sa ovog dijagrama, onda će grafički dizajnerski alat da automatski rearanžira dijagram, uključujući strelice izmedju pojedinih elemenata na dijagramu, itd.
- Npr. neki dizajnerski alati omogućuju kretanje izmedju višeg i nižeg nivoa apstrakcije.
- Zatim, grafički dizajnerski alati omogućuju proveru konzistentnosti dizajna izmedju nekoliko različitih dijagrama, a neki alati o mogućuju kreiranje programa direktno iz dizajnerskog dijagrama.

EDITORI IZVORNOG KODA U OKVIRU IDE

Neki programeri tvrde da oko 40% vremena provode editujući izvorni kod programa.

IDE (integrisano razvojno okruženje- integrated development environment) omogućuju rad sa izvornim programom (source code) sadrže u sebi editore izvornog koda. Neki programeri tvrde da oko 40% vremena provode editujući izvorni kod programa. Tako da se isplati imati dobar IDE sa dobrim editorom. Dobar IDE alat omogućuje (pored osnovnih funkcija procesiranja reči) sledeće:

- Kompilaciju i detekciju greški u okviru editora
- Integracija editora sa version-control alatima i build-alatima i debugging alatima, alatima za testiranje,
- Rezime programa tj. sažeto vidjenje programa: npr. lista imena klasa ili logička struktura programa
- Skakanje u programu na definicije klasa, podprograma i varijabli
- Skakanje na mesta gde se koriste klase, podprogrami i varijable
- Formatiranje programa
- Interaktivna pomoć
- Provera zagrada (početak-kraj blokova sa zagradama)
- Šabloni za uobičajene konstrukcije, npr. for-petlja
- Automatsko refaktorisanje
- Search-and-repace funkcija
- Search-and-replaceu nekoliko fajlova a ne samo jednom fajlu
- Editovanje nekoliko fajlova u isto vreme

- Poredjenje fajlova

REFAKTORERI (REFACTORERS) I METRIČKI REPORTERI (METRICS REPORTERS)

Refaktoreri omogućuju lake izmene programskog koda.

Refaktorer ((refactorer) tj. program za refaktorisanje, može biti samostalan (standalone) ili u okviru IDE. Refaktoreri (Refactoring browsers) pružaju sledeće mogućnosti:

- laka promena imena klase u celom programu
- laka ekstrakcija podprograma u cilju formiranja novog podprograma
- promena imena podprograma
- promena lista parametara

Dakle, refaktoreri omogućuju

- lake izmene programskog koda
- i pri tome sa manje grešaka.

Pomenimo i metričke reportere. Oni analiziraju kvalitet programa i daju izveštaj o toj analizi Oni omogućuju npr.:

- analizu kompleksnosti svakog podprograma, a u cilju dodatnog pregleda ili revizije ili ekstratestiranja ili redizajna najsloženijih podprograma
- brojanje instrukcija, deklaracija podataka, komentara, praznih linija ili u celom programu ili u pojedinim podprogramima

VERSION-CONTROL ALATI I REČNICI PODATAKA

Ovde su kratko opisani alati za upravljanje verzijama softvera, kao i tzv. rečnici podataka.

Alati za upravljanje verzijama softvera (version control tools) omogućuju:

- upravljanje verzijama programa
- upravljanjeverzijama projektne dokumentacije

Data-dictionary tj. rečnik podataka je baza podataka koja opisuje sve značajne podatke u projektu. Npr.on sadrži

- u velikim projektima stotine ili hiljade definicije klasa
- na velikim projektima omogućuje da se ima pregled imenovanja klasa itd, i izbegavanja korišćenja istih imena ili nesaglasnih imena
- ovaj alat omogućuje optimizaciju izbora imena u projektu za pojedine varijable, klase, itd

BIBLIOTEKE PROGRAMA (CODE LIBRARIES)

Preporučuje se upotreba biblioteka programa, ako je to moguće.

Vrlo je korisno u cilju skraćivanja vremena pisanja programa da se koriste već postojeći programi, ili open-source programi ili da se kupi postojeći program. U tom cilju mogu se koristiti visokokvalitetne biblioteke koje nude:

- Kontejnerske klase
- Algoritmi
- Operacije sa bazama podataka
- Matematičke operacije,
- razne vrste alata, npr. tekst alati i spelujući alati (text tools, spelling tools)
- itd.

VIDEO VEŽBA

Video vežba - Razlika između okvira i biblioteke -What is the difference between a framework and a library?

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO VEŽBA 2

Video vežba- Biblioteke otvorenog koda - How I: Use open-source libraries to make fast, happy developers.

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADACI/PITANJA ZA SAMOPROVERU ZNANJA

Pitanja za individualnu vežbu

Zadaci:

Proučiti linkove:

1) My Little UML (Tools) Page, http://plg.uwaterloo.ca/~migod/uml.html

2) Source Code Comprehension Tools, www.grok2.com/code_comprehension.html

Pitanja:

-Šta je to :

CodeSurfer, Visual Source Code Explorer (VSCE), CBrowser?

-Šta je to:
Gaphor, ArgoUML, DIA?

Poglavlje 2

Vežba - Editori za kodiranje

SYNTAX CHECKING

Syntax checking, (kad pišemo program u code-editor-u, ili kad damo komandu run) detektuje sintaksne greške, npr greške u kucanju i nemoguće izraze.

U toku rada sa programom, puno vremena se provodi u pisanju i editovanju tj uredjivanju programskog koda u Editoru programskog koda, a to je Code editor. I ako se ovaj zadatak pisanja i editovanja programskog koda čini na prvi pogled lakim i jednostavnim , i da tu ne treba neka pomoć, u stvari, pošto se tu provodi dosta vremena, i pošto programski kod može biti veliki i nepregledan i komplikovan, onda neki dodatni alati tj sposobnosti se mogu dodati u okviru Editora programskog koda koji mogu biti od velike koristi.

Ovaj editor ima nekoliko sposobnosti koje čine lakšim ovo pisanje i editovanje koda, i to

1. Syntax checking,(kad pišemo program u code-editor-u, ili kad damo komandu *run)* koji detektuje sintaksne greške, npr greške u kucanju i nemoguće izraze. Npr. ako je neki izraz problematičan, onda problematičan deo se pojavi podvučen:

.....

{

xxxxxxxxxxxxxxxxxxxxxxx;

}

Ako prevučete „miš“ iznad problematične konstrukcije, pojavi se objašnjenje u prozoru ToolTip.

GO TO DEFINITION,

Ako desno-kliknete neku varijablu ili metodu, pojavljuje se pop-up meni (Go To Definition, Outlining, itd.), koji omogućuje da „skočite“ na mesto gde se neka varijabla ili metoda deklariš

2. AutoList members, gde kada otkucate tačku u programskom kodu, pojavljuje se lista svojstava (*propereties)*, i metoda, i ovo omogućuje da se izabere željeni element.

3. Quick Info, koji pruža ToolTip informaciju o nekoj klasi ili varijabli, ako se „miš“ postavi iznad imena te klase tj varijable. Npr. ako postavite miš iznad neke varijable u nekoj instrukciji, pojaviće se informacija: o deklaraciji i tipu ove varijable. Npr. iznad varijable „yyy“:

if (xxx.yyy ! = null) {.....}

4. Go to definition, gde ako desno-kliknete neku varijablu ili metodu, pojavljuje se *pop-up* meni (*Cut, Copy, Insert Breakpoint, Go To Definition, Outlining,* itd.), koji omogućuje da „skočite“ na mesto gde se neka varijabla ili metoda deklariše.

5. Outlining**:** Ako desno-kliknete neku metodu, pojavljuje se *pop-up* meni: *Cut, Copy, Insert Breakpoint, Go To Definition, Outlining,* itd. Imamo **Outlining**, gde se omogućuje da se ako se selektuje blok teksta, onda pomoću *Hide Selection* može da se sakrije a kasnije prikaže taj blok.

Postoji Automatic Outlining, gde se svaka klasna netoda može ekspandirati ili sakriti njeno telo pomoću kliktanja simbola + i -.

AUTO-REFORMATTING

Auto-reformatting, omogućuje automatsko reformatiranje programskog koda.

6.Auto-reformatting**,** koji omogućuje automatsko reformatiranje programskog koda. Npr. ako imate:

private void xyz() {

if (.....)

{ zzzz.W.M(); }}

Posle reformatiranja imaćemo,

private void xyz()

{

if (.....)

{

zzzz.W.M();

}

}

AUTO-REFORMATTING CELOG DOKUMENTA

Može se reformatirati ceo dokument.

Dakle code-editor omogućuje automatsko reforamatiranje programa. Ovo omogućuje bolju čitljivost tj. lakše razumevanje programa, pomoću postavljanja zagrada i "inteligentnog belog prostora".

Može se reformatirati ceo dokument:

selekcija pomoću:: **Edit>Select Alll**

CLIPBOARD RING

Clipboard Ring, kod cut-copy-paste sekcija programskog koda, omogućuje da nekoliko sekcija programskog koda prikupljenih sa raznih lokacija držite zajedno na jednom mestu.

7. Clipboard Ring, kod *cut-copy-paste* sekcija programskog koda, omogućuje da nekoliko sekcija programskog koda prikupljenih sa raznih lokacija držite zajedno na jednom mestu, umesto da samo jednu sekciju programskog koda držite za potrebe *cut-copy-paste.* Clipboard Ringse nalazi uToolbox-u (sekcija na vrhu Toolbox-a)

8. Findtools**,** kojima se pristupa prekoEdit > Find and Replace meni (Find, Find in Files,Find Symbol, Replace, Replace in Files***),*** omogućuju kod velikih projekata da lako pronalazite pojedine delove projekta.

EDITORI IZVORNOG KODA U OKVIRU IDE

Neki programeri tvrde da oko 40% vremena provode editujući izvorni kod programa.

IDE (integrisano razvojno okruženje- integrated development environment) omogućuju rad sa izvornim programom (source code) sadrže u sebi editore izvornog koda. Neki programeri tvrde da oko 40% vremena provode editujući izvorni kod programa. Tako da se isplati imati dobar IDE sa dobrim editorom. Dobar IDE alat omogućuje (pored osnovnih funkcija procesiranja reči) sledeće:

- Kompilaciju i detekciju greški u okviru editora
- Integracija editora sa version-control alatima i build-alatima i debugging alatima, alatima za testiranje,
- Rezime programa tj. sažeto vidjenje programa: npr. lista imena klasa ili logička struktura programa
- Skakanje u programu na definicije klasa, podprograma i varijabli
- Skakanje na mesta gde se koriste klase, podprogrami i varijable
- Formatiranje programa
- Interaktivna pomoć
- Provera zagrada (početak-kraj blokova sa zagradama)
- Šabloni za uobičajene konstrukcije, npr. for-petlja
- Automatsko refaktorisanje

- Search-and-repace funkcija
- Search-and-replaceu nekoliko fajlova a ne samo jednom fajlu
- Editovanje nekoliko fajlova u isto vreme
- Poredjenje fajlova

JAVA CODE-EDITOR-I

Dole je dat kako izgleda jEdit editor.

Pogledati linkove:

https://mythemeshop.com/blog/code-editors/ Top 16 Code Editors for Programmers (2016)

https://blog.idrsolutions.com/2015/03/the-top-11-free-ide-for-java-coding-development-programming/ The top 11 Free IDE for Java Coding, Development & Programming

Dole je dat kako izgleda jEdit editor. Ovo je samo jedan u nizu raspoloživih editora koji se mogu naći i koji se pominje na gornja dva linka. Ovaj editor je besplatan, ali je za iskusne programere a ne početnike.

Slika 2.1 - JEdit editor za Javu za iskusne programere

JEDIT EDITOR

Evo lista osobina jEdit editora:

Evo lista osobina jEdit editora:
Prednosti:
1. Auto indentation
2. Code and text folding
3. It is the most powerful engine for implementing regular expressions
4. Spell checker, FTP support, compiler integration available using a third-party plugin
5. Multiple instances can be run at the same time
6. Integrated FTP browser

Nedostaci:
1. Being written in Java, it is a heavyweight application and thus takes time in loading
2. No collaborative editing
3. Can be buggy on Mac
4. Large files are not supported
5. SSH is not supported for remote file editing

VIDEO VEŽBA

Video vežba - Izbor Java IDE (Choosing the Right Java IDE - Java Video Tutorials).

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

INDIVIDUALNA VEŽBA

Zadaci za samostalni rad

1. Pomoću editora izvršiti automatsko formatiranje jedne Java klase
2. Pomoću editora uraditi automatsko formatiranje jedne Java klasne metode
3. Pomoću editora izvršiti automatsko formatoranje celog Java programa (cele Java aplikacije)

Poglavlje 3

CASE alati

CASE ALATI?

CASE alati automatizuju metode za dizajn, dokumentovanje i pravljenje strukrurisanog programskog koda u željenom programskom jeziku.

CASE (Computer aided software engineering) označava metode za razvoj informacionih sistema pomoću automatizovanih alata koji se mogu koristiti u toku procesa razvoja softvera. Takodje, može da označi upotrebu softvera za automatizovan razvoj softvera tj. programskog koda. CASE funkciju uključuju analizu, dizajn, i programiranje.

CASE alati automatizuju metode za dizajn, dokumentovanje i pravljenje strukrurisanog programskog koda u željenom programskom jeziku. CASE alati podržavaju aktivnosti softverskog procesa, kao što su zahtevi, dizajn, razvoj programa i testiranje. Prema tome, CASE alati obuhvataju:

Dizajn editore, Kompajlere , Debuggers (Detektori skrivenih greški*),* Itd.

CASE alati se mogu podeliti u tri grupe: Alati *(*Tools*),* Workbenches, Environments. Workbenches (radna “klupa”) i *Environments* (okruženja) su kolekcije alata. Dakle CASE tools mogu biti ili *stand alone* ili komponente u okviru *Workbenches* ili *Environments.*

CASE alati obuhvataju:

- konstrukcione alate tj. programerske alate
- requirement-specification alate
- management alate
- end-to-end-development alati
- itd

ALATI ZA KONSTRUISANJE SOFTVERA (CONSTRUCTION TOOLS)

Trenutno postojeći dizajnerski alati uglavnom su grafički alati koji prave dizajnerske dijagrame.

Sada ćemo se fokusirati na alate za konstruisanje softvera, tj. konstrukcione alate. Inače, dešava se da programeri rade po nekoliko godina pre nego što otkriju najbolje alate.

Dizajnerski alati (design tools)**:** Trenutno postojeći dizajnerski alati uglavnom su grafički alati koji prave dizajnerske dijagrame. Ovi dizajnerski alati se ponekad ugrade u CASE alat sa širim funkcijama a ne samo dizajnerskim. Grafički dizajnerski alati omogućuju da se prikaže dizajn preko UML dijagrama, arhitektonskih blok dijagrama, hijerarhijskih dijagrama, klasnih dijagrama. Neki alati se koriste samo za jednu vrstu dijagrama, a neki alati za više vrsta dijagrama. S jedne strane, ovi dizajnerski alati su samo specijalizovani paketi za crtanje. Ako se koristi obični grafički paket za crtanje ili papir i olovka, isto se može uraditi kao i pomoću ovih specijalizovanih alata, ali ovi specijalizovani alati omogućuju veću efikasnost i tačnost. Dizajnerski alat omogučuje lako pomeranje izmedju višeg i nižeg nivoa apstrakcije, jer je dizajn jedan iterativan proces na više nivoa apstrakcije. Takodje, dizajnerski alat proverava konzistentnost dizajna, i neki alati mogu da kreiraju programski kod direktno iz dizajna tj. direktno na osnovu dokumenta koji opisuje dizajn.

Alati za kodiranje izvornog programskog koda, i za debugging(source-code tools)**:** Jedna grupa alata omogućuje editovanje programskog koda. Npr. IDE obuhvata editor programskog koda. Neki programeri kažu da provode 40% vremena u pisanju tj. editovanju programskog koda. Tako da upotreba IDE je svakako opravdana. Alati za detekciju skrivenih greški, debugging, se isto nalaze u okviru IDE.

VIDEO VEŽBA

Video vežba - Softverski alati -5 Tools for Every Software Developer

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

PITANJA ZA SAMOPROVERU ZNANJA

Pitanja za individualnu vežbu

1) Koji su to konstrukcioni alati?

2)Navesti imena konkretnih konstrukcionih alata za Javu?

3)Nabrojati vrste CASE alata?

Poglavlje 4

UML alati

UML TOOLS

Postoji preko 100 UML modelujućih alata, a neki su čak besplatni.

UML alati za modelovanje su ranije bili poznati kao CASE alati. Ovi alati pomažu kod razvoja softvera kod:

- Razvoj UML dijagrama
- Kreiranje paketa, tj. organizacija UML dijagrama u pakete
- Pretraga dijagrama tj. traženje nekih odredjenih elemenata na dijagramima
- Reverzibilno inženjerstvo, tj. automatsko kreiranje klasnog dijagrama na osnovu programa, što se koristi kod modelovanja već postojećeg softvera za koji nemate UML modele
- Generisanje programskog koda, tj. automatsko kodiranje delova programa ali ne svih delova programa na osnovu UML modela

Postoji preko 100 UML modelujućih alata, a neki su čak besplatni. Oni pomažukod razvoja različitih sistema: inforamacioni sistemi, real-time iembeded sistemi, database sistemi, web-based sistemi, mobilni sistemi. Ovi UML alati omogućuju:

- Shell generation, tj. generisanje header-files na osnovu klasnog dijagrama ali bez generisanja ostalih delova programa
- Code generation, tj. kreiranje atributa klasa tj. getting/setting metoda i jednostavnih konstrukcionih metoda
- OCL (object constraint language) podrška, tj. generisanje delova programa koristeći OCL
- Podrška za kolaboraciju na velikim projektima

VIDEO VEŽBA

Video vežba - UML alat - vežba (UML Tutorial 0.2 - Installing the Eclipse Papyrus plugin for Java UML Modelling)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO TUTORIJAL

How to create a Flow Chart in Microsoft® Word 2010

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO VEŽBA ZA UML AD

Activity Diagram using Power Designer

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

VIDEO VEŽBA: UML ACTIVITY DIAGRAMS

Activity Diagrams Basic Symbols

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADACI ZA VEŽBU

Pitanja za individualnu vežbu

Zadaci za vežbu:

- Upotrebom UML alata nacrtati FC (flow-chart) dijagrame ?
- Upotrebom UML alata nacrtati AD (Activity diagram) dijagrame?
- Navesti neke UML alate koje ste koristili do sada?
- Opisati Eclipse Papyrus plugin for Java ?
- Proučiti link: My Little UML (Tools) Page, http://plg.uwaterloo.ca/~migod/uml.html ?

Poglavlje 5

Alati za testiranje i za debugging

„SOFTVERSKE SKELE“

Ove pomoćne klase ili podprogrami tj- „softverske skele“ se mogu napraviti manje ili više realistični da zamenjuju ostatak programa.

Izgradnja „ skele“ za testiranje pojedinih klasa:

„Softverske skele“ se grade da bi omogućili ispitivanje dela nekog programa tj. softvera koji se pravi iz više delova. Npr. softverska skela može biti specijalno napravljena klasa koja se može kristiti od strane druge klase koja se testira. Takva klasa se zove npr. „mock object“ ili „stub object“. Slično, može se isti prilaz koristiti kod testiranja podprograma, i tada se koriste „stub routines“. Ove pomoćne klase ili podprogrami se mogu napraviti manje ili više realistični da zamenjuju ostatak programa. Npr. oni mogu da omogućuju:

- testiranje podatka koji im se zada
- Dijagnostičke poruke, npr. „eho“ ulaznih parametara
- Prikazuje return-values za neki interaktivni ulaz
- Pojednostavljena verzija nekog objekta ili nekog podprograma

Druga vrsta softverske skele je veštački podprogram („fake routine“) koji poziva podprogram koji se testira (tzv. „test driver“), gde npr. može

- da se traže ulazni podaci da se unesu interaktivno
- da se izvrši čitav niz predefinisanih testova
- itd

TEST-DATA GENERATORS, I BAZE PODATAKA GREŠKI

Baza podataka greški, tj. Error Database, je moćan alat za testiranje.

Test-data generatori tj. random-data genberatori generišu proizvoljno tzv. test-cases tj. test-slučajeve. Npr. da proizvede 100 proizvoljnih test-slučajeva, koji omogućuju da se nađu greške u programu, npr. 1 greška za 100 test-slučajeva. Evo nekih komentara u vezi random-data generatora:

- treba dobro dizajnirati random-data generator da simulira korisnike
- ovi generatori test-slučajeva mogu da generišu neobične kombinacije podataka koje inače ne biste predvideli

- ovi generatori mogu da generišu lako ogroman broj test-slučajeva
- ove generatore treba profinjavati na osnovu iskustva, naime na osnovu rezultata testova može se videti da li je potrebno redizajnirati random-data generator

Baza podataka greški, tj. Error Database, je moćan alat za testiranje. To je baza podataka koja memoriše greške koje su bile prethodno pronađene. Ove baze podataka greški omogućuju:

- provera greški koje se ponavljaju
- praćenje korekcija greški
- itd

ALATI ZA DEBUGGING I ALATI ZA TESTIRANJE

Alati za debugging i alati za testiranje su u vezi. Alati za testiranje se mogu ili kupiti ili se može napraviti sopstveni alat.

Sledeći alati se koriste kod debugging-a:

- Test scaffolding (skele za testiranje)
- Alati za poredjenje različitih verzija programa („Diff tools“)
- Trace monitors
- Interaktivni debugger-s (Interactive Debuggers) za dijagnozu greški
- kompajleri tj. njihove poruke o greškama

A alati za testiranje pomažu kod testiranja softvera:

- Automatski test-okviri, npr.Junit, Nunit, CppUnit, itd.
- Alati za poredjenje različitih verzija programa tj. Diff Tools
- Test scaffolding ("skele za testiranje")
- Defect-tracking softver tj. baza podataka o greškama je vrlo koristan alat koji omogućuje analizu grešaka
- Test-data generatori , koji generišu npr. proizvoljno test-podatke
- diff-tools za poređenje fajlova, i poređenje izlaza programa

KOMPAJLERI, LINKERI I BUILD TOOLS

Build Tools: Cilj ovog alata je da minimizira vreme potrebno da se izgradi program koristeći trenutne verzije izvornih programskih fajlova.

Kompajleri i linkeri:

Kompajleri(compiler) konvertuju izvorni kod u izvršni kod. S druge strane, a "linker" (linker) povezuje "objektne fajlove" (object files) koje je generisao kompajler sa standardnim kodom potrebnim da se napravi izvršni program.

Build Tools:Alati za izgradnju softvera:

Ovi alati se koriste kod izgradnje velikih programa koji se sastoje od mnogo fajlova i omogućuje kombinovanje više fajlova u jednu celinu, tokom procesa izgradnje velikih programa koji se sastoje od puno fajlova. Cilj ovog alata je da minimizira vreme potrebno da se izgradi program koristeći trenutne verzije izvornih programskih fajlova. Kod modifikacije nekih fajlova u okviru velikih programa sa puno fajlova, ovi alati omogućuju brzo i pouzdano izvođenje compile-link-run ciklusa za različite verzije programa. Naime ako menjate neke fajlove u nekom velikom programu koji se sastoji od puno fajlova, npr. 200 izvornih fajlova, onda build-alati puno pomažu kod izgradnje izmenjenog programa.

VIDEO VEŽBA

Video vežba- Softverske skele (Scaffold (programming))

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

ZADACI ZA VEŽBU

Zadaci

- Napraviti jednu "softversku skelu" tipa "stubroutine"?
- Napraviti jednu softversku skelu tipa "fake routine“?

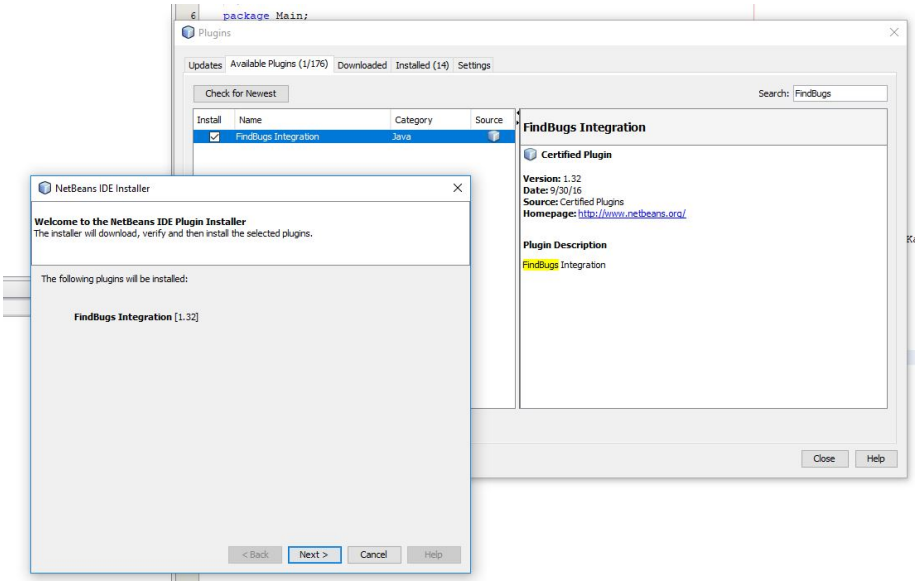
Poglavlje 6

Vežba - Alat FindBugs

POKAZNA VEŽBA- ALAT FINDBUGS

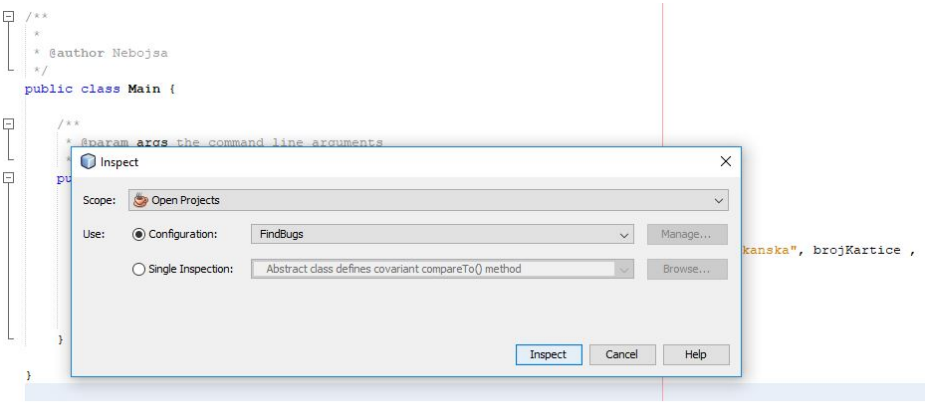
Ovde ilustrujemo alat FindBugs koji se koristi kao dodatak razvojnim okruženjima za otkrivanje grešaka u programskom kodu.

Alat FindBugs moguće je koristiti u NetBeans ili Eclipse razvojnom okruženju. Za pokretanje u okviru NetBeans razvojnog okruženja potrebno je odabrat Tools pa zatim Plugins. U okviru dostupnih dodataka potrebno je kucati FindBugs kao na slici 1:



Slika 6.1 Instalacija FindBugs alata u okviru NetBeans razvojnog okruženja

Kada je proces instalacije FindBugs dodatka završen potrebno je odabrati Source karticu u NetBeans-u, zatim opciju Inspect i podesiti konfiguraciju kao na slici 2. Potrebno je odabrati FindBugs u konfiguraciji a zatim u Scope delu odabrati "Open projects" koji znači da će FindBugs alat proveravati kompletan projekat i ceo program.

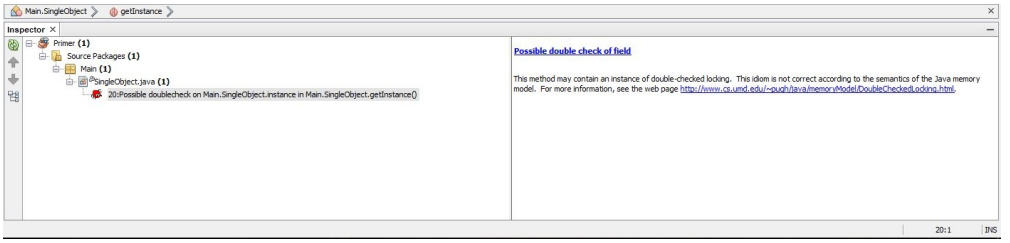


Slika 6.2 Odabir FindBugs alata u NetBeans razvojnom okruženju

POKAZNA VEŽBA- ALAT FINDBUGS - DOBIJENI REZULTATI

Rezultati pokretanja alata FindBugs prikazuju potencijalne i stvarne probleme u analiziranom programskom kodu.

Nakon pokretanja FindBugs dodatka rezultati su prikazani u formi kao na slici 3:



Slika 6.3 Dobijeni rezultati nakon pokretanja FindBugs alata u NetBeans razvojnom okruženju

Rezultati za proveru programa koji se koristi za rezervisanje karata pokazuju da postoji moguća greška i da je potrebno proveriti Main.SingleObject instancu u navedenoj putanji u programu.

Nakon izmene navedene instance i ponovnog pokretanja alata nisu pronađene potencijalne greške. Alat se može koristiti i za otkrivanje potencijalnih grešaka i za otkrivanje stvarnih grešaka u programu.

VIDEO VEŽBA

Video vežba- Alat FindBugs (FindBugs Tutorial for beginners)

Ova lekcija sadrži video materijal. Ukoliko želite da pogledate ovaj video morate da otvorite LAMS lekciju.

INDIVIDUALNA VEŽBA 1

Zadaci za individualnu vežbu.

Zadaci:

- Upoznati se detaljno sa alatom FindBugs ?
- Za izabranu aplikaciju demonstrirati primenu alataFindBugs?

DOMAĆI ZADATAK 6

Domaći zadatak 6 se dobija direktno od asistenta nakon poslatog drugog domaćeg zadatka ili na poseban zahtev studenta.

Domaći zadatak 6 je individualan za svakog studenta i dobija se direktno od asistenta.

Domaći zadatak se imenuje:

KI301-DZ06-ImePrezime-brIndeksa gde su vrednosti Ime, Prezime i br.Indeksa vaši podaci.

Domaći zadatak je potrebno poslati na adresu asistenta: nebojsa.gavrilovic@metropolitan.ac.rs sa naslovom (subject mail-a) **KI301-DZ06**.

Posebno je potrebno voditi računa o pravilnom imenovanju mail-a prilikom slanja domaćih zadataka.

Napomena: Prvi domaći zadatak je univerzalan i važi za sve studente, dok ostale domaće zadatke dobijate od asistenta nakon realizacije prethodnog.

Domaći zadaci treba da budu realizovani u razvojnom okruženju koje je definisano domaćim zadatkom i da predstavljaju jedinstveno rešenje svakog studenta. Prepisivanje i preuzimanje programskog koda sa interneta strogo je zabranjeno.

Poglavlje 7

Zaključak

ZAKLJUČAK

Evo liste nekih najvažnijih alata:

- Dizajnerski alati , Editori izvornog koda
- Alati za upravljanje verzijama (version.control tools) koji omogućuju rad sa namnoženim verzijama nekog programa
- Kompajleri i linkeri ,Biblioteke programa (code libraries)
- Debugging alati , Alati za testiranje
- Prevodioci koda (prevode program iz jednog jezika u drugi)
- Analizatori kvaliteta programa (ovi alati vrše tzv. statičku analizu programa, npr. syntax checkers, semantics checkers, metrics reporters)
- Alati za refaktorisanje (refactorers,)
- Rečnici podataka (data dictionaries): rečnik podataka u nekom projektu to je baza podataka koja sadrži opise svih značajnih podataka u projektu
- itd.

Programski alati mogu da dramatično smanje vreme konstruisanja softvera. Naime , ako koristite najnovije alate, i ako ste familijarni sa tim alatima, produktivnost može da se poveća za npr. 50%. Ovi alati takodje smanjuju onaj dosadni deo posla koji je prisutan kod programiranja, npr. neke stvari koje se ponavljaju u programima. Dobri programerski alati su vrlo korisni za programere. Ovde posmatramo samo konstrukcione alate, dok se ovde ne bavimo alatima za specifikaciju zahteva i alatima za upravljanje projektima. Neki programeri rade godinama pre nego što otkriju neke vrlo korisne alate. Cilj ovog poglavlja je da se izvrši pregled raspoloživih alata, što pomaže programerima da otkriju da li su prevideli neki alat.

LITERATURA

1. Code Complete: A practical handbook of software construction, by S. McConnell, Microsoft Press, ISBN 0-7356-1967-0, https://www.amazon.com/Code-Complete-Practical-Handbook-Construction/dp/0735619670
2. SWEBOK-V3, https://www.computer.org/web/swebok/v3
3. Theory and Problems of Software Engineering - Schaum’s Outline Series, by David Gustafson, ISBN 0-07-137794-8, http://www.mhebooklibrary.com/doi/abs/10.1036/0071377948

Linkovi:

https://www.loggly.com/blog/8-tools-for-every-java-developers-toolkit/ , Developer tools

http://www.vogella.com/tutorials/JUnit/article.html , JUnit