TDP005, Projekt: Objekt-orienterade System

Laboration: Eclipse

Höstterminen 2013

Introduktion

Detta material innehåller övningar för Eclipse.

Redovisning

Dessa övningar behöver **inte** redovisas, men ni får gärna stämma av med en assistent när ni är klara.

Utvecklingsmiljö

I kursen används Eclipse som utvecklingsmiljö och git som revisionshanteringssystem. Har ni inte git installerat sen tidigare kan det installeras med t.ex. *Synaptic* eller *apt-get*. Nedan är instruktioner för att installera Eclipse.

Installation

Installera Eclipse

Den version av Eclipse som tillhandahålls av Ubuntus pakethanteringssystem är inte alltid den senaste versionen. Därför installerar vi det manuellt istället. Dessa instruktioner beskriver processen baserat på versionen av Eclipse i skrivande stund. Vissa steg kan vara annorlunda i andra versioner.

Steg 1

Installera Eclipse manuellt:

- Ladda ner en tarboll från Eclipse hemsida:
  www.eclipse.org --> Download --> Eclipse for C++, Linux 64bit
- Packa upp filen och lägg förslagsvis mappen 'eclipse' i '~/opt/'
  (finns inte '~/opt/' så skapa den)
- Öppna mappen '~/opt/eclipse' och dra filen 'eclipse' till ditt verktygsfält högst upp på skärmen (gnome toolbar) så kan du enkelt starta Eclipse därifrån.

(*) För att kunna kompilera C++-program behöver du en kompilator för ändamålet. Installerar du endast Eclipse manuellt enligt ovan så måste du själv se till så att du har en kompilator. Installera paketet 'g++' via pakethanteringssystemet så är det fixat! Installera också 'gdb' för att kunna debugga program. Se även till så att du har paketet 'make' installerat samt 'exuberant-ctags' (för indexering).

Steg 2

Uppdatera Eclipse:

(välj en katalog för workspace – inte viktigt just nu, det går lätt att byta senare) och gå till *Workbench*. Uppdatera:

- Starta Eclipse
  - Om Eclipse inte startar pga. att en Java VM ej är installerad, installera en Java JRE via ert pakethanteringssystem. OpenJDK torde fungera, t.ex.
- Välj ett workspace, vilket som helst. Det är i denna katalog era projekt kommer lagras, i en underkatalog per projekt
- Välj *Help --> Check for updates*
- Välj *Update...*
  (vänta på ev. nedladdning)
- Starta uppdateringen (Först *Next*, sen acceptera avtal, sen *Finish*)
- Starta om Eclipse

Uppdatering klar!

Installera EGit

EGit är en git-plugin för Eclipse. Eclipse kommer numera med en Git-plugin installerat från början när man laddar ned versionen av Eclipse ovan, så installations-steget för EGit har fallit bort. Smidigt!

Användning

För att börja använda EGit tillsammans med GitLab i ett nytt projekt så gör som följer. Instruktioner inom parenteser behöver endast göras en gång:

1. (Surfa in på *http://gitlab.ida.liu.se/* och skapa ett konto.)
2. (Välj *Window --> Preferences --> General --> Network Connection --> SSH2* i Eclipse och generera ett RSA nyckelpar. Spara nyckeln lokalt i ~/.ssh/, samt kopiera den publika nyckeln och registrera den på ditt konto i GitLab. Om du redan har en RSA-nyckel i din ~/.ssh-katalog så kan du använda den istället. Kopiera isåfall innehållet från id_rsa.pub till GitLab istället.)

3. Skapa ett nytt, tomt repository på GitLab. Välj sedan *File --> Import... --> Git --> Projects from Git* i Eclipse. Välj *Clone URI* om du måste välja. Kopiera sedan in SSH-url'en från GitLab för det nya repositoryt i URI-fältet och gå vidare(se bild 1). Notera vart det lokala, motsvarande repositoryt skapas, och gå sedan vidare tills du kan välja *Use the New Project wizard(se bild 2)* och *Finish*.

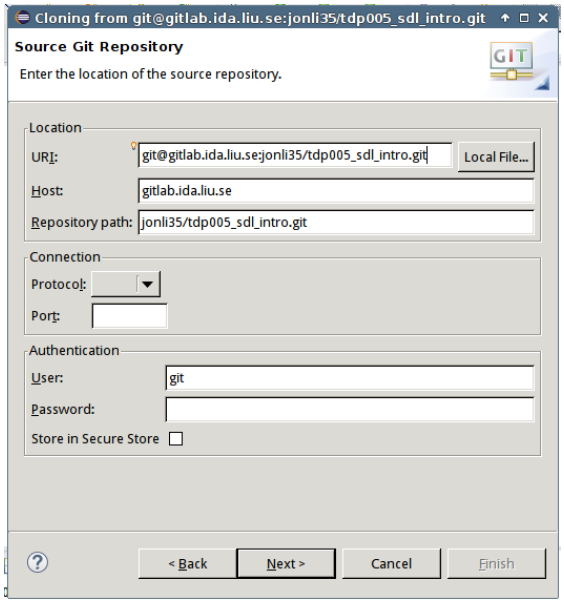

Bild 1

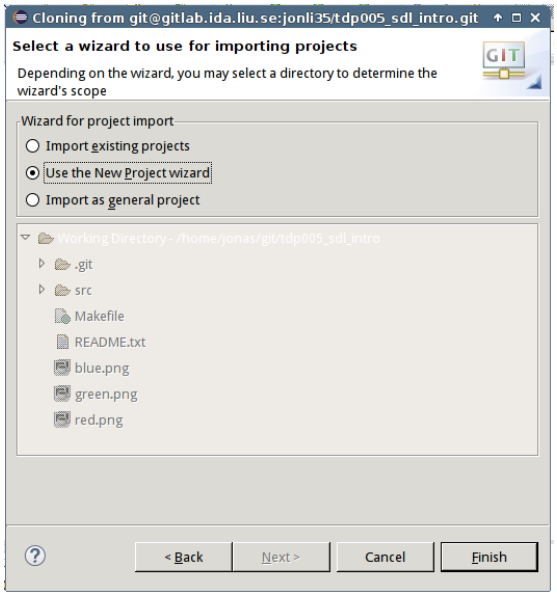

Bild 2

4. Skapa ett C++-projekt:
   C++ --> C++ Project --> *Executable* --> *Empty Project*
   Ange ett namn för projektet, och kryssa ur *Use default location* och välj där istället katalogen innuti det lokala repositoryt. Välj sedan *Finish*.

5. (Om du inte redan kan se ditt projekt i listan till vänster, öppna workbench-vyn. Om du är i *Welcome to..*-skärmen kan du hitta en knapp för detta i övre högra hörnet av Eclipse.)

6. Du kan nu högerklicka på projektet och välja *Team* för att hitta git-kommandon. Testa att välja *Commit* där för att spara dina lokala ändringar i det lokala repositoryt, och sedan *Push to Upstream* för att spara ändringarna från det lokala repositoryt till GitLab-repository. Kontrollera i GitLab att era ändringar har sparats där.
   (Det är här normalt att det dyker upp en dialog som frågar efter det namn och e-mail som du vill använda i git. Fyll i det vid behov. Du kan senare ändra dessa via *Window --> Preferences*.)

Du kan nu använda git lokalt, eller via GitLab för att dela ett repository med flera personer. GitLab kan användas utan särskilt stöd från skolans kurser, så det är ett smidigt sätt att versionshantera rent generellt. Tänk på att du manuellt måste dela med dig av dina projekt till kamrater och handledare inom GitLab för att de ska kunna nå dem. Din handledare **måste** ges tillgång till repositoryt för ditt projekt när det påbörjats!

Tidigare år har det varit en viss förvirring kring vad ett publikt och ett privat projekt på GitLab innebär. Detta påverkar endast synligheten utåt, vem som kan titta igenom koden och ladda ner den. Endast användare som ni har lagt till i projektet kan ladda upp ny kod till ert repository, oavsett synligheten i övrigt. Ni ska dock fortfarande lägga till er handledare samt kursens examinator till ert GitLab-projekt i den här kursen, så att vi enkelt kan se dessa i GitLab.

För mer information om hur git kan användas, se: http://git-scm.com/docs/gittutorial

Kom ihåg att ni kan komma åt den mesta av den här funktionaliteten via EGit i Eclipse, och det går givetvis också att använda git-kommandot från terminalen om ni navigerar er in till katalogen där det lokala repositoryt är lagrat.

Övning – Eclipse

Syftet med övningen är att ge en introduktion till Eclipse och CDT. Syftet är också att ni ska känna er bekväma med att börja använda Eclipse som editor samt att ni har tillräckligt med förståelse om Eclipse för att kunna utöka era kunskaper om verktyget på egen hand.

1. Öppna Eclipse och välj en Workspace (platsen där alla era C++-projekt ni skriver i Eclipse hamnar)
2. Första gången ni öppnar Eclipse hamnar ni i ett välkomstperspektiv. För att komma vidare till "standardperspektivet" gå vidare via ”go to the workbench” (motsvarande).
3. Skapa ett nytt C++-projekt. Detta kan du göra på flera sätt, till exempel genom att använda huvudmenyn: *File* --> *New* --> *Project*.
   Välj sedan *C++* --> *Executable* --> *Empty Project*. Du får då upp en dialogruta där du får döpa projektet. Döp det till `test1`. Övriga inställningar behöver inte ändras.
4. Börja med att göra eventuella generella inställningar för projektet. En intressant inställning kan vara att ställa in så att t.ex. c++11 används som standard. Högerklicka på projektet och välj *Properties* --> *C/C++ Build*.
5. Högst upp till höger kan du nu välja vilken typ av *Build* dessa ändringar ska gälla för. Välj *All Configurations*. Gå sedan vidare in i *Settings* --> *GCC C++ Compiler* --> *Dialect* och välj där C++11 eller nyare som *Language standard*.
   OBS: Detta är en per-projekt inställning som ni kan behöva göra om för framtida projekt.
6. Skapa en ny C++-fil genom att högerklicka på ditt projekt och välja *New* --> *Source File*.
   Döp filen till `hello.cc`.
   Skriv in koden för ett main-program som skriver ut texten "Hello world!" i filen.
7. Om något är fel i koden visar Eclipse det genom en liten röd ikon i vänsterkanten. Håll musen över ikonen för att se vad felet är. Om du inte fått några fel, prova att införa några, till exempel genom att ta bort semikolon, stava fel, inte stänga parenteser eller "glömma" inkludera paket.
   Man kan även få varningar, en liten gul ikon. Prova till exempel att deklarera en variabel som ni sedan inte använder.
8. Kör programmet.
   Det gör du till exempel genom att välja *Run* --> *Run* (Ctrl-F11) i huvudmenyn.
9. Om du vill köra ditt program igen kan du göra det genom att klicka på den lilla gröna ikonen med en pil, under huvudmenyn.
10. Det mesta i Eclipse går att göra på flera olika sätt. Prova att skapa ett nytt projekt med ett enkelt program, men utför kommandona på något annat sätt. Använd till exempel huvudmenyn om du inte gjort det innan, eller den kontextkänsliga menyn som dyker upp när man högerklickar på olika ställen i Eclipse.
11. Eclipse kan ge dig stöd på olika sätt när du programmerar. Några av dem ska vi titta på nu.
    - Deklarera en sträng i ditt program och ge den ditt namn som värde:
      ```
      string namn;
      namn = "Sara";
      ```
      Ändra i koden så att "Hello [ditt namn]" skrivs ut istället för "Hello world!" som

tidigare.
Högerklicka på variabeln `namn`. Välj *Refactor* i menyn och byt namn på variabeln till `name`.
Notera att variabelns namn byts ut på alla ställen i koden.

- Skriv i filen (inklusive punkten):
  ```
  namn.
  ```
  Då får du upp en ruta med förslag på vad som kan göras med en sträng. Titta gärna igenom alla förslag. Välj sedan `clear`. Skriv ut strängen igen. Vad gjorde `clear`?
- Skriv in bokstäverna `get` i filen. Högerklicka sedan på dem och välj *Source* --> *Content Assist* i menyn (Ctrl-Space). Då visas en lista med alla tillgängliga satser som börjar med "`get`". Titta igenom den, och välj sedan den första förekomsten av `getline`. Lägg till parametrar:
  ```
  getline(cin, name);
  ```
  Skriv ut namn igen. Vad händer när du kör programmet? Jo, det väntar på att du ska skriva in en text. Gör det! Och se vad utskriften blir.
- Prova att använda *Content Assist* på andra bokstavskombinationer och varianter. Skriv till exempel `cout << e` och tryck Ctrl-Space.

10. Det är fortfarande tidigt i kursen och objektorientering har inte kommit upp, men bara för att få en känsla för vad Eclipse kan åstadkomma ska vi prova att skapa en klass. Högerklicka på projektnamnet och välj *New* --> *Class*. Ge klassen ett godtyckligt namn. Titta på filerna som Eclipse skapar för att se vad Eclipse har genererat.
11. Ctrl-Shift-F formaterar enligt vald kodstil (vilket i kursen ska vara K&R – den förvalda). Testa att lägga in radbrytningar lite här och var mellan identifierare och måsvingar och testa sedan kortkommandot (Det blir kanske inte helt perfekt men förhoppningsvis bättre).
12. Ctrl-. (punkt) och Ctrl-, (komma) förflyttar framåt och bakåt mellan bland annat kompileringsfel i en kodfil (röda understrykningar). Kolla i *Edit* (huvudmenyn) för fler kortkommandon och även i kontextmenyn (högerklicka) under *Source*.
13. Tryck F11. Nu får du frågan om du vill byta till ett nytt perspektiv, nämligen perspektivet för debuggning. Svara ja och se hur Eclipse formar om sig till din stora ära! Öhm, jag menar: lägg märke till perspektiv-fliken och perspektiv-knappen uppe i högra hörnet av Eclipse. Där kan du byta mellan olika perspektiv.
    Om du vill kan du nu utforska debugg-perspektivet genom att till exempel stega i programmet med F5 (*step into*) eller F6 (*step over*). Fler möjligheter finns – kolla i fliken Debug uppe till vänster i Eclipse.
    Tryck F8 för att återgå till att köra programmet som vanligt. Byt sedan tillbaka till vanligt perspektiv.
14. För att städa i projektet välj *Project* --> *Cleanup...* När man städar ett projekt tas de filer bort som genereras när man bygger projektet (till exempel *.o).
15. Detta var bara ett smakprov på vad som går att göra med Eclipse. Prova gärna att göra fler saker, undersök vilka val som finns i de kontextkänsliga menyerna man får upp när man högerklickar etcetera.
16. När du känner dig klar, visa upp dina projekt för labbassistenten, och var beredd att svara på frågor kring det du gjort i övningen.

Bonus-tips:

En bra inställning att göra för era projekt kan underlätta er tid med Eclipse signifikant:

- Högerklicka på projektet, välj *Properties*
- *C/C++ Build → Behavior*
- Kryssa för *Build on resource save*

Nu kommer Eclipse automatiskt kompilera er nya kod närhelst ni sparar!

Unit-testing med googletest (Valfri övning)

Är du intresserad av enhetstestning kan den här övningen hjälpa dig komma igång med det i Eclipse. Det går givetvis att köra googletest även i terminalen, men det lämnas som en övning till läsaren.

- Starta Eclipse
- *Help → Install New Software*
- Välj *Luna* under *Work with:*
- *Programming Languages → C/C++ Unit Testing Support*
- Installera, och starta sedan om Eclipse
- Använd ert pakethanteringssystem för att installera googletest, t.ex. via *sudo apt-get install libgtest-dev*
- Skapa ett C++-projekt eller välj ett existerande att lägga till tester för
- Högerklicka på projektet och välj *New → Source Folder*, skapa en katalog döpt till *tests*
- Högerklicka den nya katalogen och välj Resource Configurations → Exclude from Build
- blabla mer från https://bjornarnelid.wordpress.com/2014/03/10/how-to-get-started-with-google-test-in-eclipse-cdt-on-linux/

Unit-testing med CUTE (Valfri övning)

Är du intresserad av enhetstestning kan den här övningen hjälpa dig komma igång med det i Eclipse. Det går givetvis att köra CUTE även i terminalen, men det lämnas som en övning till läsaren.

1. Installera CUTE med hjälp av instruktionerna från:
   http://cute-test.com/projects/cute/wiki/CUTE_Installation_and_System_Requirements
   Det går till på ungefär samma sätt som Egit installerades. Var noga med att installera rätt version av CUTE för just er version av Eclipse!
2. Skapa ett nytt projekt i Eclipse. *File --> New --> C++ Project*. Välj *CUTE --> CUTE Project*. Välj ett lämpligt projektnamn, t.ex. CUTE-labben. Klicka vidare till *Finish*.
3. Notera hur *Test.cpp* skapades, bland andra filer. Notera innehållet. Kolla upp så att du förstår vad som sker, med dokumentationen på:
   http://cute-test.com/
4. Skapa en ny klass i *src*, döp den till *Fibbonaci*. Ersätt koden för klassen i *Fibbonaci.h* med följande:

```
class Fibbonaci {
private:
	int number;
public:
	Fibbonaci(int number) : number(number) {}
	Fibbonaci();
	virtual ~Fibbonaci();

	int getNumber() const {
		return 0;
	}
};
```

5. Öppna *Test.cpp* och lägg till följande kod:

```
void fibbonaciNumberTest() {
	ASSERT_EQUALM("Fibbonaci not correct", Fibbonaci(0).getNumber(), 0);
	ASSERT_EQUALM("Fibbonaci not correct", Fibbonaci(1).getNumber(), 1);
	ASSERT_EQUALM("Fibbonaci not correct", Fibbonaci(2).getNumber(), 1);
	ASSERT_EQUALM("Fibbonaci not correct", Fibbonaci(3).getNumber(), 2);
	ASSERT_EQUALM("Fibbonaci not correct", Fibbonaci(4).getNumber(), 3);
	ASSERT_EQUALM("Fibbonaci not correct", Fibbonaci(5).getNumber(), 5);
	ASSERT_EQUALM("Fibbonaci not correct", Fibbonaci(12).getNumber(),
144);
}

void runFibbonaciSuite() {
	cute::suite s;
	s.push_back(CUTE(fibbonaciNumberTest));
	cute::ide_listener lis;
	cute::makeRunner(lis)(s, "Fibbonaci Suite");
}
```

6. Lägg till rätt includes och anropa rätt av de nya funktionerna från main-funktionen.
7. Bygg projektet, och högerklicka sedan på projektet och välj *Run As --> CUTE Test*. Du kan vara tvungen att bygga om projektet och välja *Run As*..etc igen. Om allt är rätt ska nu testet köras. Notera i den nya *CUTE Test Results*-tabben hur test-resultaten presenteras.
8. Här är det valfritt om ni vill fixa problemen som testen visar, i Fibbonaci-klassen.
9. Lägg nu till en ny fil, *main.cc*. Lägg till följande kod i den:

```
#include <iostream>
#include "Fibbonaci.h"

using namespace std;

int main(){
      cout << "Fibbonaci-talet på plats 25 är " <<
            Fibbonaci(25).getNumber() << "!" << endl;
    return 0;
}
```

10. Bygg om projektet. Nu blir det problem, eftersom det finns 2 olika main-funktioner att välja på! Det här måste fixas, om du vill ha testen i samma projekt som det färdiga programmet. Du måste ställa in så att en main används för att generera test-programmet, och den andra main-funktionen för att generera det ”riktiga” programmet.
    Lägg till:
    *#ifndef TEST_*
    <main-funktionen>
    #endif
    ..kring main-funktionen i *main.cc*, och:
    *#ifdef TEST_*
    <main-funktionen>
    #endif
    ..kring main-funktionen i *Test.cc*. Notera att den ena innehåller *#if**n**def*, den andra *#ifdef*. Dessa förändrar vilken kod som kommer med i kompileringen beroende på om symbolen *TEST_* är definierad eller ej.
11. Högerklicka på projektet och välj *Properties --> C/C++ Build* och välj *Manage Configurations* till höger. Klicka *New...*, döp den till *Test Debug* och välj *Copy settings from --> Existing configuration --> Debug*. Klicka *Ok*. Stäng *Manage Configurations*-rutan och välj *Test Debug* högst upp, och välj sedan *C/C++ Build --> Settings --> Preprocessor* till vänster och lägg till *TEST_* som symbol. Klicka *Ok*.
12. Nästan klar. Högst uppe i menyn, välj *Run --> Run Configurations.. --> <projekt-namn>*. Ställ in *Build configuration:* till *Test Debug* där nu. Ändra även *C/C++ Application:* till *Test Debug/CUTE*-labben. Ändra även *Name:* till *CUTE-labben Test Suite.* Välj *Apply* och stäng rutan. Bygg nu om projektet, och gå upp i *Run* och kör om testet. Nu ska testet fungera att köra igen.
13. Högerklicka nu på projektet, välj *Run As --> Local C/C++ Application*. Du kan vara tvungen att bygga om projektet och göra det igen. Nu ska det ”vanliga” programmet köras, och du ska nu kunna använda både testet och det ”vanliga” programmet från samma projekt, så länge som du kör precis det testprogrammet som du nyss gjorde inställningarna för via *Run*-menyn.

14. Njut!