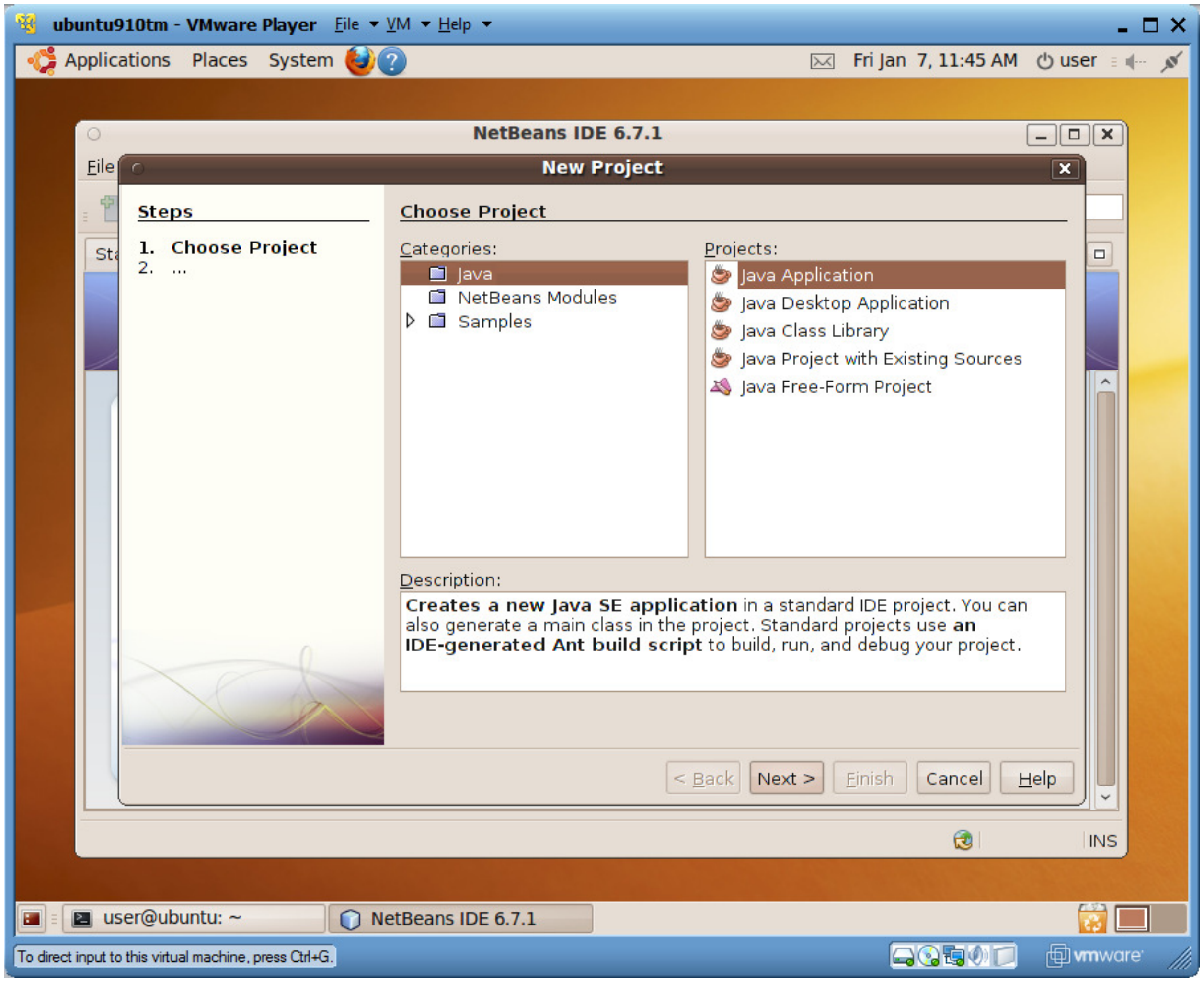
CPSC 2800 Linux Hands-on Lab # 2 Developing Linux Application in Java, C and C++

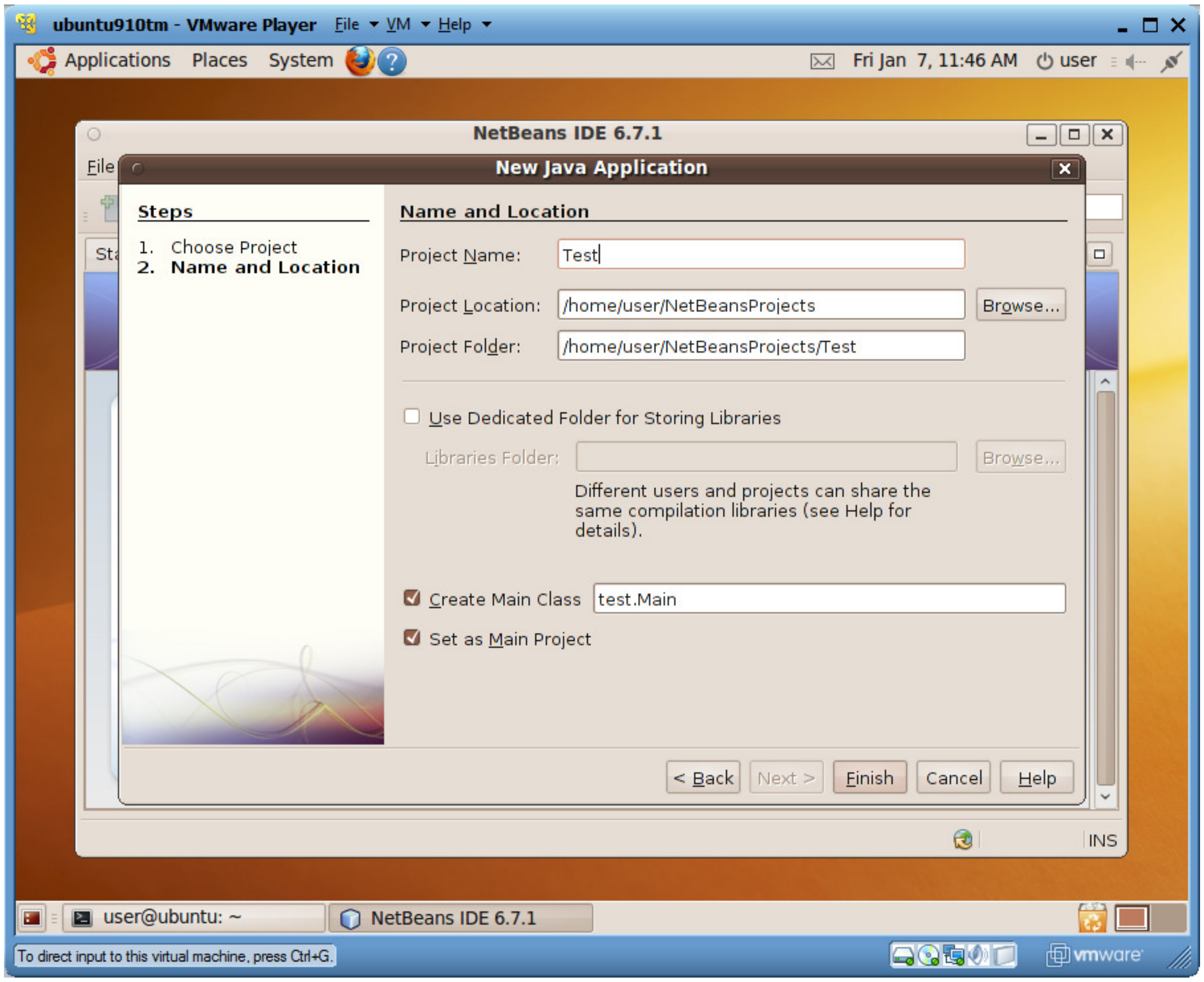
Project 2-1 Develop a Java test program in NetBeans

- If you have not installed Netbeans IDE yet, install it by typing **sudo apt-get install netbeans** in a terminal window.
- Enter root password “123456” for old version of Linux or “12345678” for new version of Linux.
- And type **Y** when it is asked.
- If you have not launched *NetBeans* IDE yet, start it by typing command **netbeans** in a terminal window.
- Click on menu item “**File|New Project …**” to launch the “**New Project**” window. Make sure *Categories* selection chooses “**Java**”, and Projects selection chooses “**Java Application**”, as shown below.



Creates a new Java SE application in a standard IDE project. You can also generate a main class in the project. Standard projects use an IDE-generated Ant build script to build, run, and debug your project.

- Click on the *Next* button and see the “**New Java Application**” window. Type “**Test**” in the “**Project Name**” text field, as shown below. Make sure that the checkboxes for “**Create Main Class**” and “**Set as Main Project**” are checked.



Name and Location

Project Name: Test

Project Location: /home/user/NetBeansProjects Browse...

Project Folder: /home/user/NetBeansProjects/Test

☐ Use Dedicated Folder for Storing Libraries

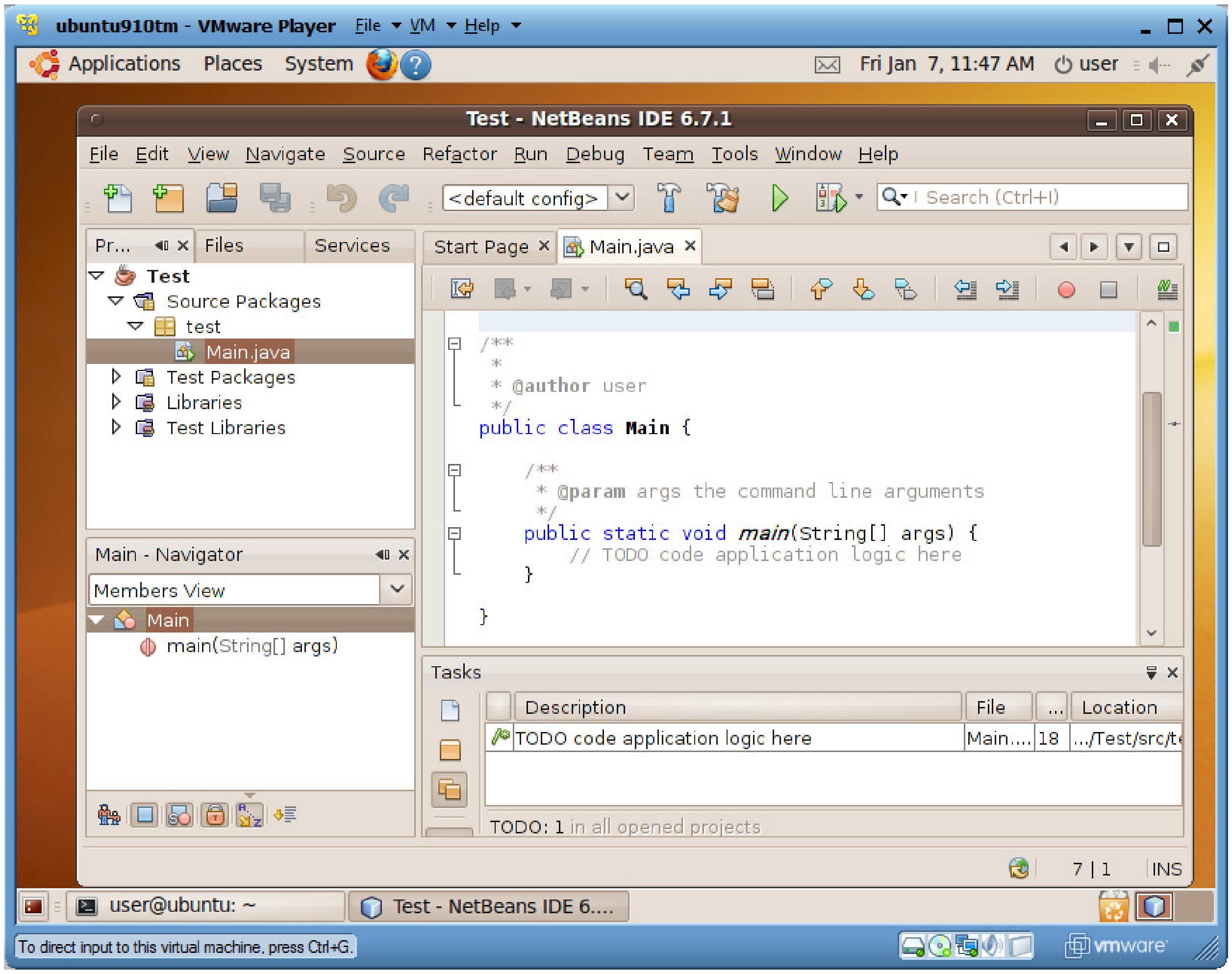
Libraries Folder: Browse...

Different users and projects can share the same compilation libraries (see Help for details).

☑ Create Main Class test.Main

☑ Set as Main Project

- Click on the **Finish** button, and you will see a screen similar to the following one:

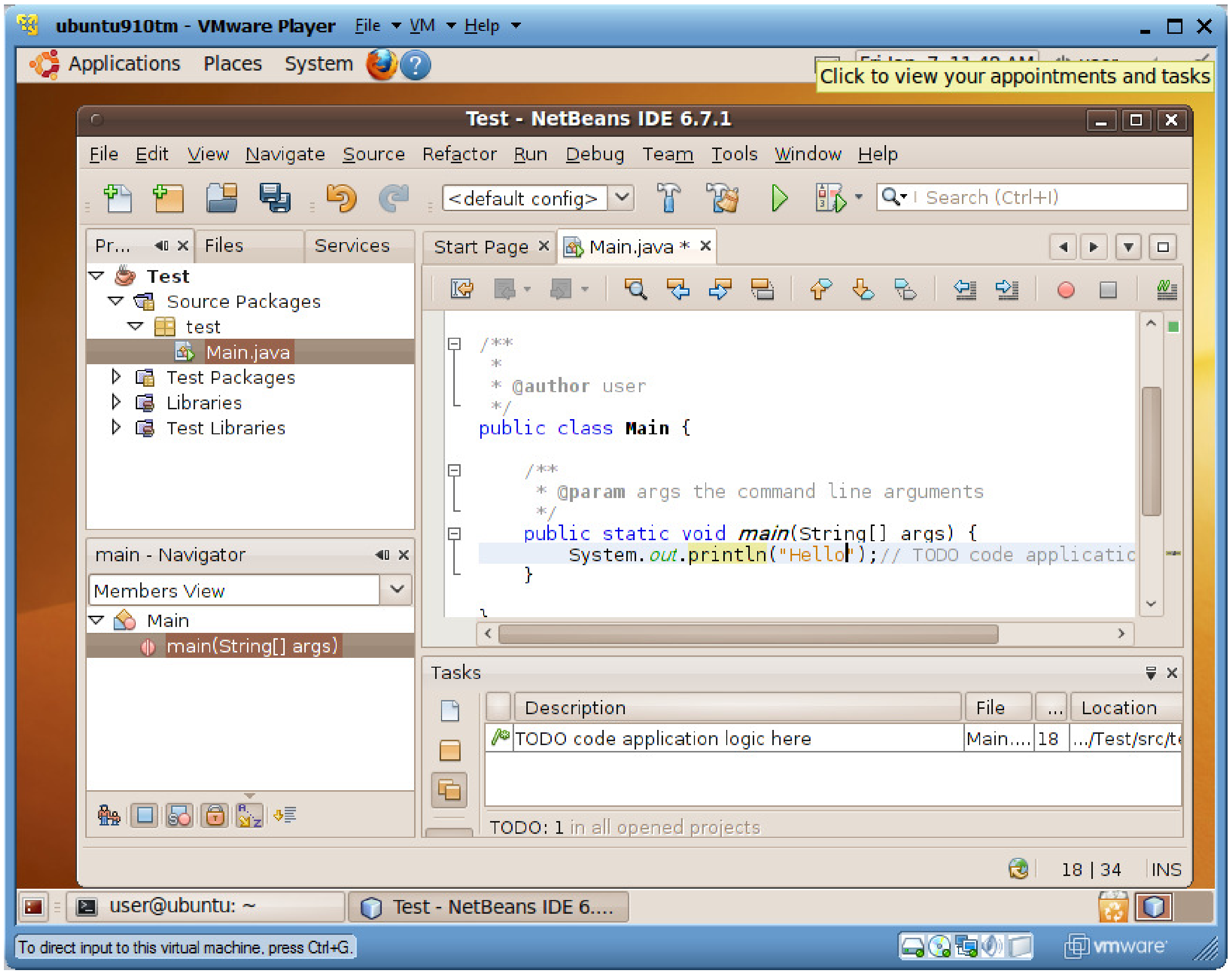


```
/**
 *
 * @author user
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}
```

	Description	File	...	Location
	TODO code application logic here	Main....	18	.../Test/src/t

TODO: 1 in all opened projects

- In the body of method main, enter “**System.out.println(“Hello”);**”, as shown below.

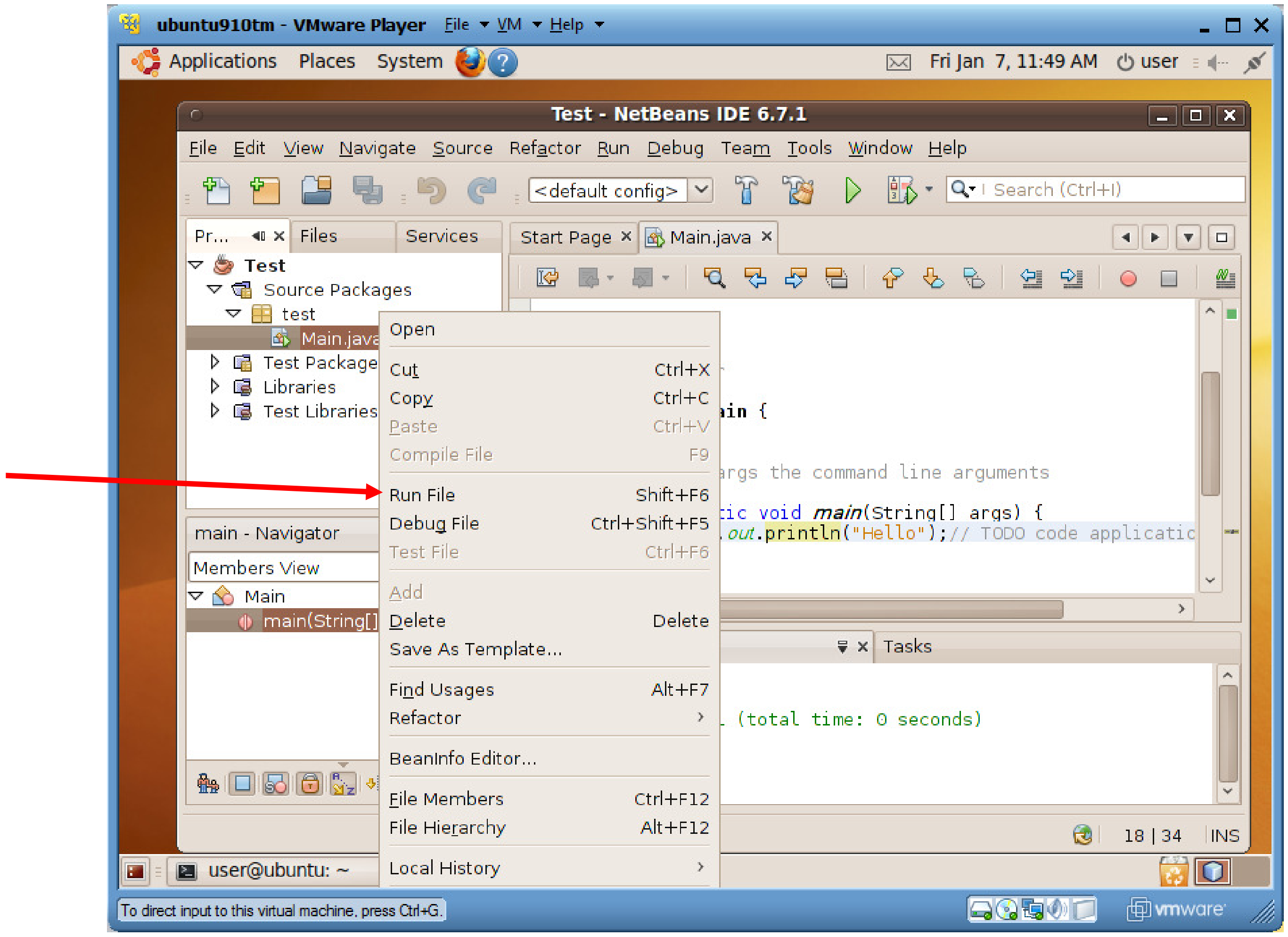

ubuntu910tm - VMware Player
Applications Places System
Click to view your appointments and tasks
Test - NetBeans IDE 6.7.1
File Edit View Navigate Source Refactor Run Debug Team Tools Window Help
<default config>
Search (Ctrl+I)
Pr... Files Services
Test
Source Packages
test
Main.java
Test Packages
Libraries
Test Libraries
main - Navigator
Members View
Main
main(String[] args)
Start Page
Main.java *
Tasks
18 | 34
INS
user@ubuntu: ~
Test - NetBeans IDE 6....
To direct input to this virtual machine, press Ctrl+G.
vmware

```
/**
 *
 * @author user
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello");// TODO code applicatic
    }
```

	Description	File	...	Location
	TODO code application logic here	Main....	18	.../Test/src/t

TODO: 1 in all opened projects

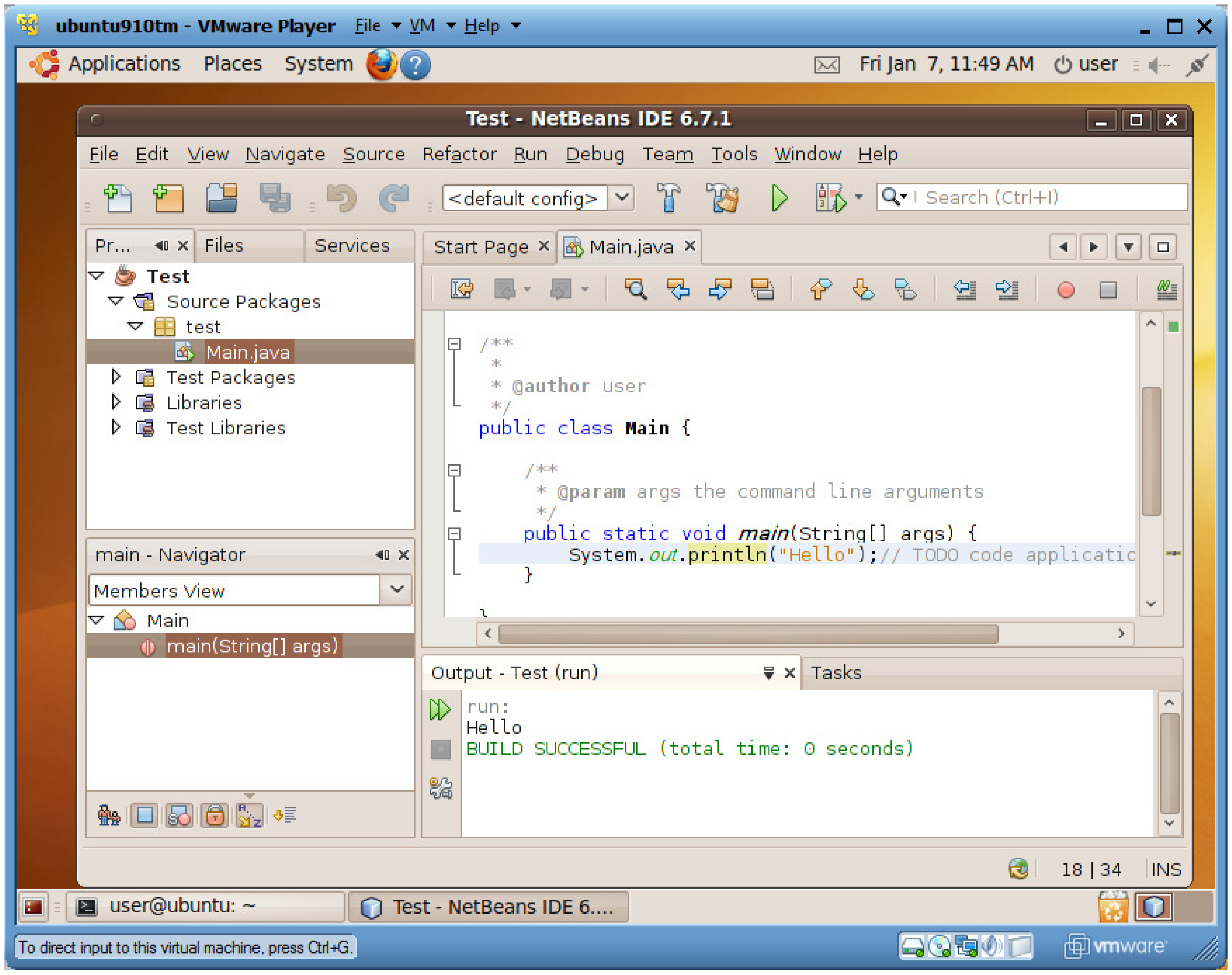
- In the left-upper project pane, right-click on file “**Main.java**” and choose “**Run File**” on the popup menu, as show below:



```
ain {

args the command line arguments

tic void main(String[] args) {
.out.println("Hello");// TODO code applicatic
```

```
. (total time: 0 seconds)
```

- The program will be saved to hard disk, compiled into a bytecode file, and executed. The execution output is displayed under the “Output” tab, as shown below:



```
/**
 *
 * @author user
 */
public class Main {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello");// TODO code applicatic
    }
```

```
run:
Hello
BUILD SUCCESSFUL (total time: 0 seconds)
```

- Click on menu item "**File|Close Project**" to close this project, and use "**File|Exit**" to shut down *NetBeans* IDE.
- Your *NetBeans* projects are saved under "/home/user/NetBeansProjects".

Project 2-2

In this project you create and run a simple C program.

To write a simple C program:

1. Use the vi or other editor (such as gedit or nedit) to create and save the file **inchines.c** (Remember that the C complier uses the .c extension to identify a file containing C source code). Enter this code:

   /* this program converts 10 feet to inches. */

   #include <stdio.h>

   int main(){

   int inches, feet;

```
        feet = 10;

        inches = feet * 12;

        printf("There are %d inches in %d feet. \n", inches, feet);

}
```

```
/* this program converts 10 feet to inches. */
#include  <stdio.h>
#include <unistd.h>
int main(){
        int pid, i;
        setbuf(stdout,NULL);
        pid = fork();
        if(pid == 0) {
                printf("Hey! \n");
        }
        else if (pid < 0){
                fprintf(stderr, "Could not fork!\n");
        }
        else {
                printf("Hello!\n");
                for(i=0; i<60; i++){
                        sleep(1);
                        printf(".");
                }
                printf("Bye!\n");
        }
        return 0;
}
```

-

2. Save the program and exit the editor.
3. The C compiler is executed by the gcc command in Linux. Type **gcc inches.c** and press **Enter**. If you typed the program correctly, you see no messages. If you see error messages, load the program into editor, and correct the mistake.
4. By default, the compiler stores the executable program in a file named a.out. Execute a.out by typing **./a.out** and press **Enter**. Record your screen: ______________________________________
5. You can specify the name of the executable file with the –o option. Type **gcc –o inches inches.c** and press **Enter**. The command complies the inches.c file and stores the executable code in a file named inches.
6. Run the inches program by typing **./inches** and pressing **Enter**.
7. Type **clear** and press **Enter** to clear the screen for the next project.

Project 2-3

In this project, you create a C program that uses an if-else statement.

To use the C if-else statement:

1. Create the file **radius.c** with your choice of editor. Enter the following C code:

```
#include <stdio.h>
int main(){
        float radius = 50, area;
        area = 3.14159 * radius * radius;
        if (area > 100)
                printf("The area, %f, is too large. \n", area);
        else
                printf("The area, %f, is within limits. \n", area);
}
```

2. Save the file and exit the editor.
3. Compile the program by typing **gcc –o radius radius.c** and pressing **Enter**. If you see error messages, edit the file, and correct your mistakes.
4. Execute the program by typing **./radius** and pressing **Enter**. Record your output: ______________
   ____
5. Type **clear** and press **Enter** to clear the screen.

Project 2-4
In this project, you create a C program using a for loop.
To practice using a C for loop:

1. Use the editor of your choice to create the file **rain.c**, entering this C code:

```
/* rain.c*/
#include <stdio.h>
int main(){
        int rain, total_rain=0;
        for (rain=0; rain< 10; rain++)
        {
                printf("We have had %d inches of rain. \n", rain);
                total_rain = total_rain + rain;
        }
        printf("We have had a total ");
        printf("of %d inches of rain. \n", total_rain);
}
```

2. Save the file and exit the editor.
3. Compile the program and store the executable code in a file named **rain**.
4. Run the program. Your screen look like: ____
5. Type **clear** and press **Enter** to clear the screen.

Project 2-5
Functions can be powerful tools in C programming. In this program, you create two functions that accept arguments and return a value.

To practice writing functions that accepts arguments and returns a value:

1. Use the editor of your choice to create the file **absolute.c**, entering this C code:

```
#include <stdio.h>
int absolute(int num);
int main(){
        int x = -12, y;
        y = absolute(x);
        printf("The absolute value of %d is %d\n", x, y);
}

int absolute(int num)
{
        if (num < 0)
                return (-num);
        else
                return (num);
}
```

2. Save the file and exit the editor.
3. Compile the program and store the executable code in a file named absolute.
4. Run the program. Your screen look like: ____
5. Type **clear** and press **Enter** to clear the screen.

Project 2-6

In this project, you create a C program that performs file input and output.

To perform input/output:

1. Use the editor of your choice to create the **file buildfile.c**. Enter the following code in the file:

```
#include <stdio.h>

int main(){
        FILE *out_file;
        int count = 0;
        char msg[] = "This was created by a C program. \n";
        if((out_file = fopen("testfile", "w")) == NULL)
        {
                printf("Error opening file. \n");
                return(1);
        }
        while(count < 33)
        {
                fputc(msg[count], out_file);
                count++;
        }
        fclose(out_file);
}
```

2. Save the file and exit the editor.
3. Compile the program and save the executable in a file named **buildfile**.
4. Run the **./buildfile** program. The program creates another file, testfile.

5. To see the contents of testfile, type cat testfile and press Enter. Record your output: ___________
   ____
6. Type **clear** and press **Enter** to clear the screen.

Project 2-7
In this project, you create two files and link them together into one executable file.
To compile and link to files:

1. Use the editor of your choice to create the file **abs_func.c** and enter the following code:
```
int absolute(int num)
{
        if (num < 0)
                return (-num);
        else
                return (num);
}
```
2. Save the file.
3. Create the file **abs_main.c**. Enter this code:
```
#include <stdio.h>
int absolute(int num);
int main(){
        int x = -12, y;
        y = absolute(x);
        printf("The absolute value of %d is %d\n", x, y);
}
```
4. Save the file and exit the editor.
5. Compile and link the two programs by typing **gcc abs_main.c abs_func.c –o abs** and then press **Enter**. The compiler separately compile abs_main.c and abs_func.c. Their object files are linked together, and the executable code is stored in the file abs.
6. Run the **./abs** program. Record your output: ______________________________________
7. Type **clear** and press **Enter** to clear the screen.

Project 2-8
In this project, you use the make utility and a makefile to create a multimodule C project.
To crate a simple multimodule C project.

1. Use the editor of your choice to create the file **square_func.c** and enter the following code in the **file:**
```
int square(int number)
{
        return (number * number);
}
```
2. Save the file.
3. Next crate the file **square_main.c**. Enter the following code:
```
#include <stdio.h>

int main(){
        int count, sq;
        for (count = 1; count < 11; count++)
        {
```

```
                        sq = square(count);
                        printf("The square of %d is %d \n", count, sq);
                }
        }
```

4. Save the file.
5. Next create a makefile named **make_square**. Enter the following text:
   square_func.o: square_func.c
   (press Tab) gcc -c square_func.c
   square_main.o: square_main.c
   (Tab) gcc -c square_main.c
   square: square_func.o square_main.o
   (Tab)gcc square_func.o square_main.o -o square
6. Save the file and exit the editor.
7. Build the program by typing **make –f make_square square** and pressing **Enter**.
8. Run the program by typing **./square**. Record your output:___________________________________
9. Type **clear** and **Enter** to clear the screen.

Project 2-9
In this project you create a simple C++ program.
To write a C++ program:
1. If you do not have C++ complier (i.e., g++) installed, type **sudo apt-get install g++** in a terminal window.
2. Enter root password "**123456**" for old version of Linux or "12345678" for new version of Linux.
3. And type **Y** when it is asked.
4. Use the editor of your choice to create the simple.C file. Enter the following code:

```
//============================================================
//Program Name:       Simple.C
//By:                 your initials
//Purpose:            Firsst program in C++ showing how to produce output
//============================================================
#include <iostream>
using namespace std;
int main(void)
{
        cout << "C++ is a programming language. \n";
        cout << "Like C, C++ is compatible with UNIX/Linux. \n";
}
```

5. Save the simple.C and exit the editor.
6. Use the C++ compiler to create a programm called sim_plus by typing **g++ simple.C –o sim_plus** and then press **Enter**.
7. Run sim_plus by typing **./sim_plus**. What does your screen look like? _________________________ _______
8. Type **clear** and **Enter** to clear the screen.

Project 2-10

Reading a file is important for C++ programming. In this project, you create a C++ program that reads the contents of a file.

To create a C++ program that reads a text file:

1. Use the editor of your choice(such as **gedit**) to create the **fileread.C** file. Enter the following code:

```
//=================================================================
//Program Name:         fileread.C
//By:                   your initials
//Purpose:              A C++ program reads the contents of a file
//=================================================================
#include <iostream>
#include <fstream>
using namespace std;
int main(void)
{
        ifstream file("testfile");
        char record_in[256];
        if (file.fail())
                cout << "Error opening file. \n";
        else
        {
                while(!file.eof())
                {
                        file.getline(record_in, sizeof(record_in));
                        if(file.good())
                                cout << record_in <<endl;
                }
        }
}
```

2. Save the file and exit the editor.
3. Type **g++ fileread.C –o fileread**
4. Test the program by typing **./fileread**. Record your output: ______________________________

   ____
5. Type **rm testfile** to remove testfile.
6. Test the program again by typing **./fileread**. Record your output: ______________________________

   ____
7. Type **clear** and **Enter** to clear the screen.

Include your experiences and answers to all the underlying parts in your report. Include the following at the beginning of your report.

- Name: ______________________________
- UTC ID: ______________________________
- Course Number and Name: ______________________________
- Semester: ______________________________
- Lab Name and Number: ______________________________

- I spent _______________hours and ____________minutes to finish this hands-on lab.
- I have _______________ (percent) finish this lab.
- I expect ______________ (A, B, C, or F) of this lab.
- This lab helps me to master Java, C, and C++ compilation and execution under Linux Operating System and environment. Choose a number to indicate how much the lab is helpful.

  1 2 3 4 5

  (less helpful) (more helpful)