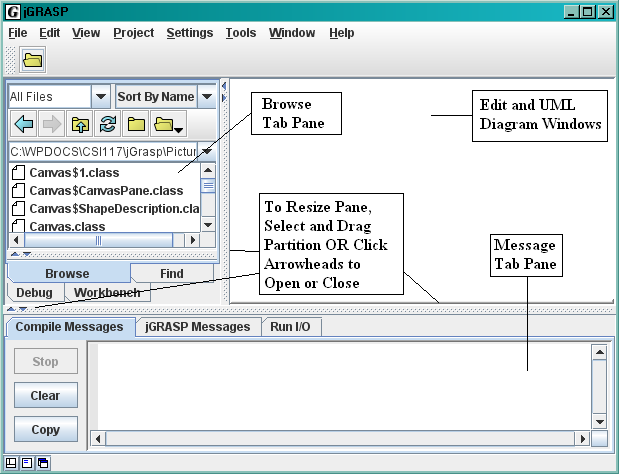
**Programming in Java Using jGrasp**

* 1. **Getting Started**

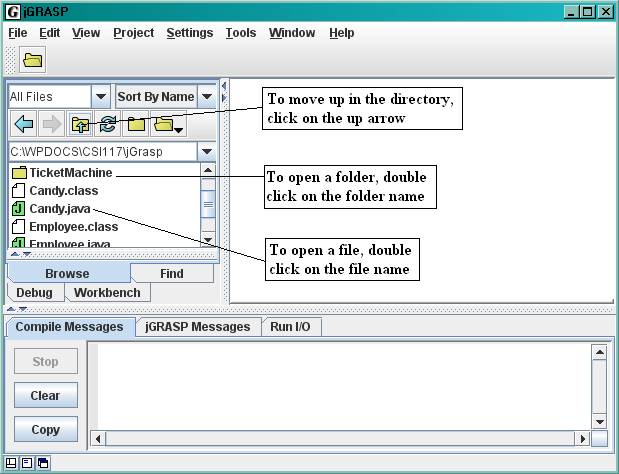
To start jGrasp, double click on the jGrasp icon on the Desktop, or go to the Start menu and choose jGrasp. Depending on your computer’s speed, jGrasp may take as long as 30 seconds to start up.

The jGrasp window is shown in the figure below. It consists of a menu and toolbar at the top, and 3 resizable panes. The *left pane* contains the following tabs: 1) **Browse** – lists files in the current directory, 2) **Debug** – used when debugging a program, 3) **Find** – searches a program, and 4) **Workbench** – creates instances of classes and invokes methods. The large *right pane* is used to display UML diagrams and to edit a program. The *lower pane* has tabs for compile messages, jGrasp messages, and input/output when running a program.

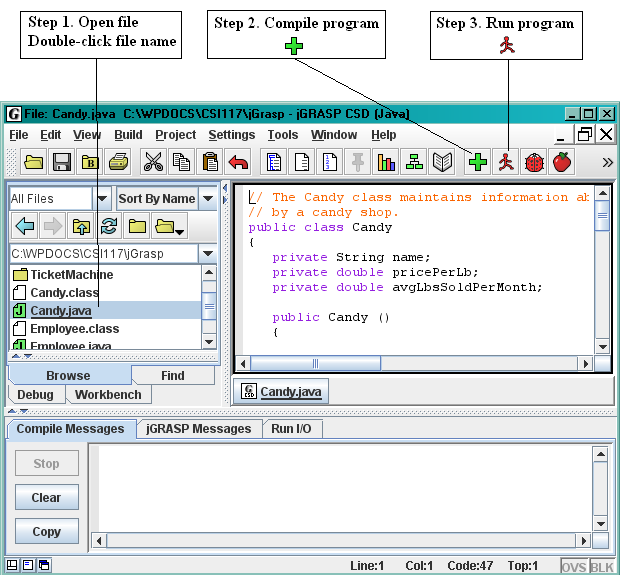
****

**1.2 Open a File**

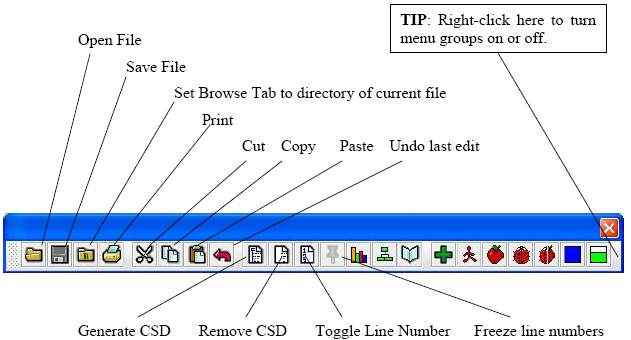
The easiest way to open an existing program is to use the **Browse** tab, as shown in the figure below. The UP arrow goes up one level, the LEFT arrow goes back to the previous directory, the RIGHT arrow goes forward, the R refreshes the Browse pane, and the H shows hidden files. To open a directory or a file, double-click on it.

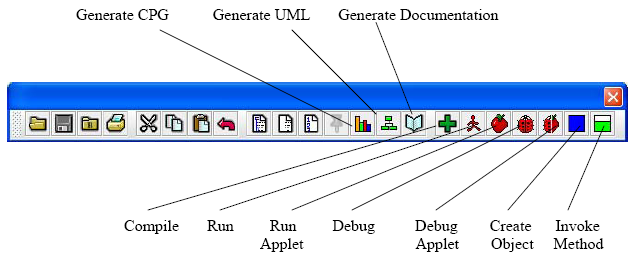


**1.3 Compiling and Running a Program**



**1.4 Other Menu Buttons**





**Activity 1**

***Perform the following steps:***

* Execute jGrasp.
* Create a new Java file by clicking on File/New/Java. Then choose File/Save As and name the file HelloWorld. Make sure the file has the .java extension.
* Type the following Java code. Be sure to follow the indentation guidelines and include the comments.

/\*\*

\* The HelloWorld class displays "Hello, world!" on the screen.

\* @author

**Type your own name here.**

\*/

class HelloWorld

{

public static void main(String[] args)

{

System.out.println ("Hello, world!"); // display Hello, world

return;

}

}

* Save the file.
* Compile the program. If you have errors, correct them and then compile again.
* Once the program compiles without errors, Run the program. You should see the program output.
* Experiment with creating errors. For instance, remove one of the semicolons, compile, and then look at the error message. Put the semicolon back in and remove one of the curly braces. Compile and look at the error this causes. This is a good exercise to try on a simple program so that you can see the error messages given by the compiler.

**Activity 2**

***Perform the following steps:***

* Using the HelloWorld.java file you created in Activity1, save the file as MyHelloWorld.java
* Locate the line class HelloWorld  
  Change HelloWorld to MyHelloWorld   
  because the name of the .java file and the name of the class must always be the same.
* Locate the instruction: System.out.println ("Hello, world!");

**Type your own name here.**

Add the following lines after that instruction:

System.out.println ("My name is: ");

System.out.println ("CSI 117");

* Compile the program. If you have errors, correct them and then compile again.
* Once the program compiles without errors, Run the program. You should see the program output.

**Activity 3**

***Perform the following steps:***

* Choose File/Close All to close all your currently open files.
* Create a new Java file by clicking on File/New/Java. Then choose File/Save As and name the file Student. Make sure the file has the .java extension.
* Type the following Java code. Be sure to follow the indentation guidelines and include the comments.

/\*\*  
 \* Student class  
 \*   
 \*/  
public class Student  
{  
 /\*\*  
 \* score for first test  
 \*/   
 private int test1;  
  
 /\*\*  
 \* score for second test  
 \*/   
 private int test2;  
  
 /\*\*  
 \* student’s average test score  
 \*/   
 private double average;  
  
 /\*\*  
 \* Default constructor   
 \*/  
 public Student()  
 {  
 test1 = 0;  
 test2 = 0;  
 average = 0;  
 return;  
 }  
  
 /\*\*  
 \* Overloaded constructor   
 \*/  
 public Student(int newTest1, int newTest2)  
 {  
 setTest1(newTest1);  
 //setTest2(newTest2);  
 calcAverage();  
 return;  
 }

/\*\*  
 \* the getTest1 method returns the value of the test1 attribute  
 \*/  
 public int getTest1()  
 {  
 return test1;  
 }  
  
 /\*\*  
 \* the setTest1 method gives a value to the test1 data member  
 \*/  
 public void setTest1(int newTest1)  
 {  
 if (newTest1 >= 0 && newTest1 <= 100)  
 test1 = newTest1;

else

{

System.out.println (newTest1 + " is an invalid value for test 1.");

test1 = 0;

}  
 return;  
 }  
  
 /\*\*  
 \* the getAverage method returns the average   
 \*/  
 public double getAverage( )   
 {  
 return average;  
 }  
   
 /\*\*  
 \* the calcAverage method calculates the average  
 \*/  
 public void calcAverage( )   
 {  
 average = (test1 + test2) / 2.0;  
 return;  
 }  
   
 /\*\*  
 \* the displayGrade method displays an appropriate letter grade   
 \*/  
 public void displayGrade( )   
 {  
 System.out.println ("Test 1 score is: " + test1);   
 System.out.println ("Test 2 score is: " + test2);   
 if (average >= 90)  
 System.out.println("A");  
 else if (average >= 80)  
 System.out.println("B");  
 else if (average >= 70)   
 System.out.println("C");  
 else if (average >= 60)   
 System.out.println("D");  
 else   
 System.out.println("F");  
 }  
   
}

* Save the file.
* Compile the class. If you have errors, correct them and then compile again.
* Add the accessor and mutator method for the test2 attribute. Then remove the // in front of the //setTest2(newTest2); statement in the overloaded constructor; i.e., uncomment the call to the setTest2() method. Make sure the program compiles before continuing.
* Create a new Java file by clicking on File/New/Java. Then choose File/Save As and name the file StudentTest. Make sure the file has the .java extension.
* Type the following Java code. Be sure to follow the indentation guidelines and include the comments.

/\*\*  
 \* The StudentTest class creates Students, calculates their average,   
 \* and displays their letter grade.  
 \* @author Type your name here  
 \*/  
class StudentTest  
{   
 public static void main(String[] args)   
 {  
 Student john = new Student();   
 Student jill = new Student(96, 88);  
   
 // set test scores for john, calculate his average, and   
 // display his grade  
 john.setTest1(83);  
 john.setTest2(75);  
 john.calcAverage();  
 john.displayGrade();  
 System.out.println();  
   
 // display Jill's grade. Scores were set and average was   
 // calculated by the constructor.  
 jill.displayGrade();  
 return;   
 }  
}

* Compile the class and correct any errors.
* Once the program compiles without errors, Run the program. You should see the program output.

**2.0 Candy Exercise**For help with Java, read the PseudocodeToJava document on the course web site.

1. Execute jGrasp.
2. Choose File/Close All to close all the currently open files.
3. Choose File/New/Java to begin a new Java class in the edit window.
4. Type the following code into the edit window:

// The Candy class maintains information about candy sold  
// by a candy shop.  
public class Candy  
{  
 private String name;  
 private double pricePerLb;  
 private double avgLbsSold;  
  
 public Candy ()  
 {  
 name = " ";  
 pricePerLb = 0;  
 avgLbsSold = 0;  
 return;  
 }  
  
 public Candy (String newName, double newPricePerLb, double newAvgLbsSold)  
 {  
 name = newName;  
 //setPricePerLb (newPricePerLb);  
 //setAvgLbsSold (newAvgLbsSold);  
 return;  
 }  
  
 public String getName()  
 {  
 return name;  
 }  
  
 public void setName (String newName)  
 {  
 name = newName;  
 return;  
 }  
}

1. Save the file by choosing File/Save As, and using the name Candy.java. Make sure the class compiles before continuing.
2. Add two methods called getPricePerLb() and setPricePerLb() that will return the value of pricePerLb and set the value of pricePerLb, respectively. Remember that the data type of pricePerLb is double. Then remove the // in front of the //setPricePerLb (newPricePerLb); statement in the overloaded constructor; i.e., uncomment the call to the setPricePerLb() method. Make sure the class compiles before continuing.
3. Add two methods called getAvgLbsSold() and setAvgLbsSold () that will return the value of avgLbsSold and set the value of avgLbsSold, respectively. Remember that the data type of avgLbsSold is double. Then remove the // in front of the //setAvgLbsSold (newAvgLbsSold); statement in the overloaded constructor; i.e., uncomment the call to the setAvgLbsSold() method. Make sure the class compiles before continuing.
4. Choose File/New/Java to begin a new Java class in the edit window. Create a new class named CandyShop, saving it in a file named CandyShop.java. Add the main() program to the CandyShop class, creating two Candy objects. Name the first Candy object fudge and use the default constructor. Name the second Candy object taffy and use the overloaded constructor (the second constructor), making up appropriate values for the parameters. Call each of the mutator methods for the fudge object, providing appropriate values for the parameters. Write 3 more Java statements that call each of the 3 accessors for the fudge object, placing the calls inside an appropriate Java print statement that identifies what is being printed. When you call the constructor that initializes all the data members, or you call the setName() method, make sure you surround the name of the candy with quotation marks (" ").

**3.0 Employee Exercise**

1. Choose File/Close All to close all your currently open files.
2. Choose File/New/Java to begin a new Java class in the edit window.
3. Using the steps in the Candy Exercise as a model, translate the pseudocode for the classes below into Java and create the classes in Java using jGrasp. Take note that the payRate, hoursWorked and grossPay are type double in Java.

Class Employee

Private Real payRate

Private Real hoursWorked

Private Real grossPay

Public Module Employee ( )

Set payRate = 0.0

Set hoursWorked = 0.0

Set grossPay = 0.0

Return

Public Function Real getPayRate ( )

Return payRate  
 End Function

Public Function Real getHoursWorked ( )

Return hoursWorked

End Function

Public Function Real getGrossPay ( )

Return grossPay

End Function

Public Module setPayRate (Real newPayRate)

If newPayRate > 0 Then

Set payRate = newPayRate

Else

Display newPayRate, " is not a valid pay rate."

Set payRate = 0

End If

End Module

Public Module setHoursWorked (Real newHoursWorked)

If newHoursWorked > 0 Then

Set hoursWorked = newHoursWorked

Else

Display newHoursWorked, " is not a valid value for hours worked."

Set hoursWorked = 0

End If

End Module

Public Module computeGrossPay ( )

Set grossPay = payRate \* hoursWorked

Display "Gross pay is", grossPay

End Module

End Class

Module main()

Declare Employee julie  
 Set julie = New Employee()

Call julie.setPayRate (18.75)

Call julie.setHoursWorked (47.5)

Call julie.computeGrossPay ( )

End Module

1. After you have translated the design into a Java program, compile both classes. Correct any syntax errors before continuing.
2. Run the PayProgram class.