**(Prerequisite: CIS 406)**

**COURSE DESCRIPTION**

This course covers advanced topics in the Java object-oriented programming language. Students will test, document, and design business-oriented programs and solve advanced programming problems.  Topics include advanced data structures, recursion, multithreading, and the application of Java constructs to the Internet and database development.

**INSTRUCTIONAL MATERIALS**

**Required Resources**

Farrell, J., (2014). *Java Programming* (7th ed.). Independence, KY: Cengage.

Eclipse Integrated Development Environment (IDE) is recommended by Strayer University to its students and faculty as the preferred IDE for this course. Further information on the free download of the Eclipse IDE, alternative IDEs, restrictions, and tutorials can be found under *Software* in the Notes section of this Course Guide.

**Supplemental Resources**

Eckel, B. (2002). *Thinking in Java, 3rd edition*. Upper Saddle River, NJ: Prentice Hall.

Java Beginner. (2008). General format. Retrieved from <http://www.javabeginner.com/>

The Java Tutorials. (2012). General format. Retrieved from <http://docs.oracle.com/javase/tutorial/>

**COURSE LEARNING OUTCOMES**

1. Discuss the concepts of inheritance and polymorphism.
2. Compare and contrast abstract and concrete data types.
3. Discuss the fundamentals of exception handling.
4. Apply the Java concepts of file input / output.
5. Design programs that handle events from user-interface components.
6. Analyze the methods used to implement graphics in Java programs.
7. Examine the implementation of Applets, multimedia, and sound in Java programs.
8. Examine the method of recursion.
9. Compare and contrast multithreading and single-threaded processes.
10. Use technology and information resources to research issues in Java programing.
11. Write clearly and concisely about Java programing using proper writing mechanics and technical style conventions.

**WEEKLY COURSE SCHEDULE**

The standard requirement for a 4.5 credit hour course is for students to spend 13.5 hours in weekly work. This includes preparation, activities, and evaluation regardless of delivery mode.

|  |  |  |
| --- | --- | --- |
| **Week** | **Preparation, Activities, and Evaluation** | **Points** |
| 1 | Preparation   * Reading(s)   + Chapter 10: Introduction to Inheritance * Tutorial   + The students should access and use the following tutorials prior to completing the lab assignments for this lesson. A set of tutorials and information on the use of the Eclipse Integrated Development Environment (IDE) is located at <http://eclipsetutorial.sourceforge.net/totalbeginner.html>**.**   + Further information on the free download of the Eclipse IDE, alternative IDEs, restrictions, and tutorials can be found under *Software* in the Notes section of this Course Guide. * Video(s), accessible in the online course shell   + “Inheritance” (2 min 52 s)   + “Handling Methods and Inheritance” (3 min 59 s)   + “Constructors and Inheritance” (3 min 12 s)   Activities   * Discussion   Evaluation   * Lab 1: Exercise 1 on page 539 * Lab 2: Exercise 8 on page 541 | 20  20  20 |
| 2 | Preparation   * Reading(s)   + Chapter 11: Advanced Inheritance Concepts * Video(s), accessible in the online course shell   + “Abstract Classes” (3 min 28 s)   + “The Object Class” (3 min 37 s)   + “Interfaces” (4 min 21 s)   Activities   * Discussion   Evaluation   * Lab 3: Exercise 9 on page 597 * Lab 4: Exercise 13 on page 598 | 20  20  20 |
| 3 | Preparation   * Reading(s)   + Chapter 12: Exception Handling * Video(s), accessible in the online course shell   + “Exceptions” (1 min 55 s)   + “Catching Multiple Exceptions” (3 min 24 s)   + “Specifying Exceptions” (5 min 55 s)   Activities   * Discussion   Evaluation   * Lab 5: Exercise 4 on page 667 * Lab 6: Exercise 8 on page 668 | 20  20  20 |
| 4 | Preparation   * Reading(s)   + Chapter 13: File Input and Output * Video(s), accessible in the online course shell   + “Paths and Attributes” (2 min 56 s)   + “File Organization, Streams, and Buffers” (5 min 45 s)   + “Random Access Data Files” (2 min 45 s)   Activities   * Discussion   Evaluation   * Lab 7: Exercise 3 on page 735 * Lab 8: Exercise 5 on page 735 | 20  20  20 |
| 5 | Preparation   * Reading(s)   + Chapter 14: Introduction to Swing Components * Video(s), accessible in the online course shell   + “Using the Jframe class” (2 min 27 s)   + “Using a Layout Manager” (2 min 19 s)   + “Event-Driven Programming” (5 min 29 s)   Activities   * Discussion   Evaluation   * Midterm Exam: Chapters 10 through 13 | 20  50 |
| 6 | Preparation   * Reading(s)   + Chapter 15: Advanced GUI Topics * Video(s), accessible in the online course shell   + “Layout Managers” (1 min 41 s)   + “The JPanel Class” (2 min 32 s)   + “Event Handling” (5 min 56 s)   Activities   * Discussion   Evaluation   * Assignment 1: Insurance Agent App: Part 1 | 20  180 |
| 7 | Preparation   * Reading(s)   + Chapter 16: Graphics * Video(s), accessible in the online course shell   + “Using paint() and repaint()” (4 min 23 s)   + “Drawing Lines and Shapes”’ (4 min 18 s)   + “Font Methods” (6 min 03 s)   Activities   * Discussion   Evaluation   * Lab 9: Exercise 1 on page 936 * Lab 10: Exercise 3 on page 937 | 20  20  20 |
| 8 | Preparation   * Reading(s)   + Chapter 17: Applets, Images, and Sound * Video(s), accessible in the online course shell   + “Writing and Running an Applet” (5 min 32 s)   + “Working with JApplet Components” (3 min 55 s)   + “Using Images and Sound in an Applet” (4 min 34 s)   Activities   * Discussion   Evaluation   * Case Study: Bar Chart Java | 20  100 |
| 9 | Preparation   * Reading(s)   + Dale, N., Joyce, D., & Weems, C. (2002). O*bject-oriented data structures using Java*. Sudbury, Massachussetts: Jones and Bartlett Publishers.   Chapter 7: Programming with Recursion. Retrieved from <http://cdn.preterhuman.net/texts/math/Data_Structure_And_Algorithms/Object-Oriented%20Data%20Structures%20Using%20Java%20-%20Nell%20Dale.pdf>   * Video   + Watch the video titled, “Intermediate Java Tutorial - 3 - Recursion” (8 min 37 s). Be prepared to discuss.   Video Source: thenewboston. (2009, Dec 20). Intermediate Java Tutorial - 3 - Recursion [Video file]. Retrieved from <http://www.youtube.com/watch?v=fpuWkZs51aM>  This video can be viewed from within your online course shell.  Activities   * Discussion   Evaluation   * Lab 11: Exercise 9a on page 525 of the e–Book provided * Lab 12: Exercise 15a on page 526 and 527 of the e–Book provided | 20  20  20 |
| 10 | Preparation   * Reading(s)   + Schildt, H. (2008). *Herb Schildt's Java Programming Cookbook*. New York, NY: McGraw-Hill.   Chapter 7: Multithreading. Retrieved from <http://libdatab.strayer.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=nlebk&AN=214920&site=eds-live&scope=site&ebv=EB&ppid=pp_295>   * Video   + Watch the video titled, “Intermediate Java Tutorial - 26 - Learning about Threads” (7 min 41 s). Be prepared to discuss.   Video Source: thenewboston. (2010, May 3). Intermediate Java Tutorial - 26 - Learning about Threads [Video file]. Retrieved from <http://www.youtube.com/watch?v=VYN-CBtPNiM>  This video can be viewed from within your online course shell.  Activities   * Discussion   Evaluation   * Assignment 2: Insurance Agent App: Part 2 | 20  180 |
| 11 | Preparation   * Reading(s): None   Activities   * Discussion   Evaluation   * Final Exam: Chapters 14 through 17 | 50 |

**GRADING SCALE – UNDERGRADUATE**

|  |  |  |
| --- | --- | --- |
| **Assignment** | **Total Points** | **% of**  **Grade** |
| Lab Participation (12 labs worth 20 points apiece) | 240 | 24% |
| Assignment 1: Insurance Agent App: Part 1 | 180 | 18% |
| Case Study: Bar Chart Java | 100 | 10% |
| Assignment 2: Insurance Agent App: Part 2 | 180 | 18% |
| Midterm Exam (Chapters 10-13)  (open book, 2-hour time limit, with 25 multiple choice questions worth 2 points apiece) | 50 | 5% |
| Final Exam (Chapters 14-17)  (open book, 2-hour time limit, with 25 multiple choice questions worth 2 points apiece) | 50 | 5% |
| Participation (10 discussions worth 20 points apiece)  **Note:** Week 11 discussion is not graded. | 200 | 20% |
| Totals | 1,000 | 100% |

|  |  |  |
| --- | --- | --- |
| **Points** | **Percentage** | **Grade** |
| 900 – 1,000 | 90% – 100% | A |
| 800 – 899 | 80% – 89% | B |
| 700 – 799 | 70% – 79% | C |
| 600 – 699 | 60% – 69% | D |
| Below 600 | Below 60% | F |

**Writing Assignments**

The objective of the School of Information Systems’ writing assignments is to promote attitudes and skills that will improve a student’s ability to communicate in writing, develop research skills and documentation techniques, and encourage critical analysis of data and conclusions specific to the course learning outcomes in the information systems and technology domain.

**Lab Assignments**

Worth 20 points apiece

Complete the weekly labs based on the following:

* Write the code for each lab assignment.
* The lab is to be submitted in a single zip file in the online course shell, which must contain all .java files, along with any additional files that may be necessary for your project to run (ex: text files).
* Any and all written answers must be entered into the online course shell with the submission of the attached lab assignment.

|  |  |
| --- | --- |
| **Weekly Lab Breakdown** | |
| **Week Due** | **Graded Lab Exercises** |
| 1 | Lab 1: Exercise 1 on page 539  Lab 2: Exercise 8 on page 541 |
| 2 | Lab 3: Exercise 9 on page 597  Lab 4: Exercise 13 on page 598 |
| 3 | Lab 5: Exercise 4 on page 667  Lab 6: Exercise 8 on page 668 |
| 4 | Lab 7: Exercise 3 on page 735  Lab 8: Exercise 5 on page 735 |
| 5 | None |
| 6 | None |
| 7 | Lab 9: Exercise 1 on page 936  Lab 10: Exercise 3 on page 937 |
| 8 | None |
| 9 | Lab 11: Exercise 9a on page 525 of the e–Book provided  Lab 12: Exercise 15a on page 526 and 527 of the e–Book provided |
| 10 | None |
| 11 | None |

Each lab assignment will be graded based on the following:

1. The program must compile, execute, produce correct results, and meet all of the specifications in the weekly lab.

Additionally you must:

1. Organize the code for user readability.
2. Organize the code for reusability.
3. Provide documentation with embedded comments for reader understanding.
4. Organize the code for efficiency.

Grading for each lab assignment will be based on the following rubric.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Points: 20** | **Java Programming Lab Assignment Rubric** | | | | |
| **Criteria** | **Unacceptable**  **Below 60% F** | **Meets Minimum Expectations**  **60-69% D** | **Fair**  **70-79% C** | **Proficient**  **80-89% B** | **Exemplary**  **90-100% A** |
| 1. Specifications  Weight: 60% | The program does not compile. | The program compiles but does not execute. | The program compiles and executes but produces incorrect results. | The program compiles, executes, and produces correct results but does not meet all of the specifications. | The program compiles, executes, produces correct results, and meets all of the specifications. |
| 2. Readability Weight: 10% | The code is not organized and very difficult to read. | The code is poorly organized and difficult to read. | The code is partially organized but readable only by someone who knows the expected end result. | The code is organized and easy to read. | The code is exceptionally organized and very easy to read. |
| 3. Reusability and object-oriented programming constructs  Weight: 10% | The code is not organized for reusability. | The code is poorly organized for reusability. | The code is partially organized and some parts of the code could be reused in other programs. | The code is organized and most of the code could be reused in other programs. | The code is exceptionally organized and could be reused as a whole or each routine could be reused. |
| 4. Documentation  Weight: 10% | No documentation is provided. | The documentation consists of embedded comments but does not help the reader understand the code. | The documentation consists of embedded comments and some header comments separating routines. | The documentation consists of embedded comments and header documentation that is useful in understanding the code. | The documentation consists of embedded comments and clearly explains what the code is accomplishing and how. |
| 5. Efficiency  Weight: 10% | The code is unnecessarily long and appears to be patched together. | The code is unnecessarily long. | The code is fairly efficient but sacrifices readability and understanding. | The code is efficient without sacrificing readability and understanding. | The code is extremely efficient without sacrificing readability and understanding. |

**Assignment 1: Insurance Agent App: Part 1**

Due Week 6 and worth 180 points

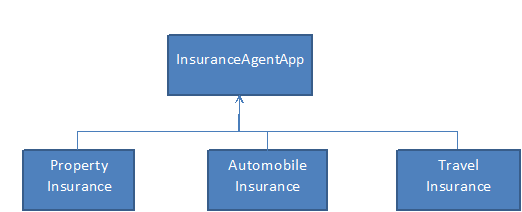
This assignment consists of two (2) sections:

* A Java program file
* A screenshot of the driver testing and source code of the implementation

Label each file name according to the section of the assignment for which it is written. Put both sections together in a single zip file, and submit the file.

It is a common knowledge that most households, for protection against their major perils, take out insurance on their property, automobiles, travel, and so on. Insurance premiums are often paid on a monthly basis. In this assignment, you will build an interactive Java application that will help an insurance agent generate a quote for a household. Assume that the household wants to purchase property insurance, automobile insurance, and travel insurance. The Java application must generate a quote when the agent runs the application.

The application must contain four (4) Java classes that are stated and shown in the diagram below.



The **InsuranceAgentApp** class is an abstract class. The **PropertyInsurance**, **AutomobileInsurance**, , and **TravelInsurance** classes inherit from the **InsuranceAgentApp** abstract class.

The specifications of each class are described below. The column on the left shows the class members and the column on the right provides a description of the class members. The ‘**S’** designates a *static* member; the ‘**C’** designates a constructor, and the ‘**A’** designates an *abstract* member. The data types of each class members are shown on the left column.

|  |  |
| --- | --- |
| **Class Members and Their Properties** | **Member’s Description** |
|  | * **type** is the type of insurance (e.g. “Property”). * **Name** is the name of the insured object (e.g. “Home”); **totalPremium** is the *static* quote for all the insurances taken by the household. * **InsuranceAgentApp(String)** is the constructor that takes the **type** value. * three *get* methods for the three variables. * **InsuranceAgentApp()** is the empty constructor; **setInsuredObjectName**(), **setRiskAmount()**, and **display()** are abstract methods. |
|  | * **riskAmount** is the value of the Property being insured. * the static **RATE** is the % used to calculates the annual premium. * The constructor **PropertyInsurance(String)** takes a string which indicates the insurance type (e.g. “Property”). * the three *get* methods return values of the three variables. * **setInsuredObjectName**(), **setRiskAmount()**, and **display()** are implementations of the abstract methods. |
|  | * **riskAmount** is the value of the Automobile being insured. * the static **RATE** is the % used to calculates the annual premium. * The constructor **AutomobileInsurance(String)** takes a string which indicates the insurance type (e.g. “Automobile”). * the three *get* methods return values of the three variables. * **setInsuredObjectName**(), **setRiskAmount()**, and **display()** are implementations of the abstract methods. |
|  | * **riskAmount** is the value of the flight being insured; the static **RATE** is the % used to calculates the annual premium. * The constructor **TravelInsurance(String)** takes a string which indicates the insurance type (e.g. “Travel”). * the three *get* methods return values of the three variables. * **setInsuredObjectName**(), **setRiskAmount()**, and **display()** are implementations of the abstract methods. |

The RATEs used to calculate the premiums are:

|  |  |  |
| --- | --- | --- |
| PropertyInsurance | AutomobileInsurance | TravelInsurance |
| 0.25% | 0.75% | 0.73% |

**Section 1: Java Program File**

Create an interactive Java application for the insurance agent in order to generate a quote for a household. According to your design, the program must:

* Implement all the four (4) Java classes described with the correct inheritance of abstract classes.
* Calculate the premium and multiply the value of the object being insured by the corresponding RATE. **Note**: See the RATE tables above.
* Get a quote and add all the premiums for all the objects being insured.
* Include a driver problem to test the implementation in which it:
  + Uses the **System.out.\*** and **System.in.\*** methods for all the input / output.
  + Includes an object of each insurance type. Prompts the user to enter the **type** and **riskAmount** (or value of the object being insured).
  + Displays the values in each of the objects created using the **display()** method.
  + Displays the quote (**totalPremium** in the abstract **InsuranceAgentApp** class).

**Section 2: Screenshot of the Driver Testing and Source Code of the Implementation**

Create a screenshot of the driver testing and include a copy of source code of the implementation.

* Submit a screenshot of the driver testing of the implementation of your Java program. **Note**: Go to <http://www.take-a-screenshot.org/> if you need a tutorial on taking a screenshot.
* Submit a copy of the source code of the implementation.

**Section 1 and Section 2 will be graded based on the following:**

1. The program must compile, execute, produce correct results, and meet all of the specifications stated in Section 1.

Additionally you must:

1. Organize the code for user readability.
2. Organize the code for reusability.
3. Organize the code for efficiency.
4. Provide documentation with embedded comments for reader understanding.

The specific course learning outcomes associated with this assignment are:

* Discuss the concepts of inheritance and polymorphism.
* Compare and contrast abstract and concrete data types.
* Apply the Java concepts of file input / output.
* Use technology and information resources to research issues in Java programing.

Grading for this assignment will be based on the following rubric.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Points: 180** | **Assignment 1: Insurance Agent App: Part 1** | | | | |
| **Criteria** | **Unacceptable**  **Below 60% F** | **Meets Minimum Expectations**  **60-69% D** | **Fair**  **70-79% C** | **Proficient**  **80-89% B** | **Exemplary**  **90-100% A** |
| 1. Specifications  Weight: 60% | The program does not compile. | The program compiles but does not execute. | The program compiles and executes but produces incorrect results. | The program compiles, executes, and produces correct results but does not meet all of the specifications. | The program compiles, executes, produces correct results, and meets all of the specifications. |
| 2. Readability Weight: 10% | The code is not organized and very difficult to read. | The code is poorly organized and difficult to read. | The code is partially organized but readable only by someone who knows the expected end result. | The code is organized and easy to read. | The code is exceptionally organized and very easy to read. |
| 3. Reusability and object-oriented programming constructs  Weight: 10% | The code is not organized for reusability. | The code is poorly organized for reusability. | The code is partially organized and some parts of the code could be reused in other programs. | The code is organized and most of the code could be reused in other programs. | The code is exceptionally organized and could be reused as a whole or each routine could be reused. |
| 4. Efficiency  Weight: 10% | The code is unnecessarily long and appears to be patched together. | The code is unnecessarily long. | The code is fairly efficient but sacrifices readability and understanding. | The code is efficient without sacrificing readability and understanding. | The code is extremely efficient without sacrificing readability and understanding. |
| 5. Documentation  Weight: 10% | No documentation is provided. | The documentation consists of embedded comments but does not help the reader understand the code. | The documentation consists of embedded comments and some header comments separating routines. | The documentation consists of embedded comments and header documentation that is useful in understanding the code. | The documentation consists of embedded comments and clearly explains what the code is accomplishing and how. |

**Case Study: Bar Chart Java**

Due Week 8 and worth 100 points

Read the article titled, “Teaching software componentization: A bar chart Java bean,” located in the online course shell. (<http://libdatab.strayer.edu/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=bth&AN=58034299&site=eds-live&scope=site>)

Write a two to four (2-4) page paper in which you:

1. Summarize the main points of the article.
2. Describe one (1) way in which one could apply Java graphics in education. Provide one (1) example of such use to support your response.
3. Analyze at least three (3) commands that you would use in order to draw a bar chart. Provide a rationale for your response.
4. Suggest one (1) way in which you would use to modify a bar chart in order to update it for new information. Justify your response.

Your assignment must follow these formatting requirements:

* Be typed, double spaced, using Times New Roman font (size 12), with one-inch margins on all sides; citations and references must follow APA or school-specific format. Check with your professor for any additional instructions.
* Include a cover page containing the title of the assignment, the student’s name, the professor’s name, the course title, and the date. The cover page and the reference page are not included in the required assignment page length.

The specific course learning outcomes associated with this assignment are:

* Analyze the methods used to implement graphics in Java programs.
* Use technology and information resources to research issues in Java programing.
* Write clearly and concisely about Java programing using proper writing mechanics and technical style conventions.

Grading for this assignment will be based on answer quality, logic / organization of the paper, and language and writing skills, using the following rubric.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Points: 100** | **Case Study: Bar Chart Java** | | | | |
| **Criteria** | **Unacceptable**  **Below 60% F** | **Meets Minimum Expectations**  **60-69% D** | **Fair**  **70-79% C** | **Proficient**  **80-89% B** | **Exemplary**  **90-100% A** |
| 1. Summarize the main points of the article.  Weight: 10% | Did not submit or incompletely summarized the main points of the article. | Insufficiently summarized the main points of the article. | Partially summarized the main points of the article. | Satisfactorily summarized the main points of the article. | Thoroughly summarized the main points of the article. |
| 2. Describe one (1) way in which one could apply Java graphics in education. Provide one (1) example of such use to support your response. Weight: 30% | Did not submit or incompletely described one (1) way in which one could apply Java graphics in education. Did not submit or incompletely provided one (1) example of such use to support your response. | Insufficiently described one (1) way in which one could apply Java graphics in education. Insufficiently provided one (1) example of such use to support your response. | Partially described one (1) way in which one could apply Java graphics in education. Partially provided one (1) example of such use to support your response. | Satisfactorily described one (1) way in which one could apply Java graphics in education. Satisfactorily provided one (1) example of such use to support your response. | Thoroughly described one (1) way in which one could apply Java graphics in education. Thoroughly provided one (1) example of such use to support your response. |
| 3. Analyze at least three (3) commands that you would use in order to draw a bar chart. Provide a rationale for your response.  Weight: 25% | Did not submit or incompletely analyzed at least three (3) commands that you would use in order to draw a bar chart. Did not submit or incompletely provided a rationale for your response. | Insufficiently analyzed at least three (3) commands that you would use in order to draw a bar chart. Insufficiently provided a rationale for your response. | Partially analyzed at least three (3) commands that you would use in order to draw a bar chart. Partially provided a rationale for your response. | Satisfactorily analyzed at least three (3) commands that you would use in order to draw a bar chart. Satisfactorily provided a rationale for your response. | Thoroughly analyzed at least three (3) commands that you would use in order to draw a bar chart. Thoroughly provided a rationale for your response. |
| 4. Suggest one (1) way in which you would use to modify a bar chart in order to update it for new information. Justify your response.  Weight: 25% | Did not submit or incompletely suggested one (1) way in which you would use to modify a bar chart in order to update it for new information. Did not submit or incompletely justified your response. | Insufficiently suggested one (1) way in which you would use to modify a bar chart in order to update it for new information. Insufficiently justified your response. | Partially suggested one (1) way in which you would use to modify a bar chart in order to update it for new information. Partially justified your response. | Satisfactorily suggested one (1) way in which you would use to modify a bar chart in order to update it for new information. Satisfactorily justified your response. | Thoroughly suggested one (1) way in which you would use to modify a bar chart in order to update it for new information. Thoroughly justified your response. |
| 5. Clarity, writing mechanics, and formatting requirements  Weight: 10% | More than 8 errors present | 7-8 errors present | 5-6 errors present | 3-4 errors present | 0-2 errors present |

**Assignment 2: Insurance Agent App: Part 2**

Due Week 10 and worth 180 points

This assignment consists of two (2) sections:

* Java program file(s)
* A screenshot of the execution and source code of your Java program

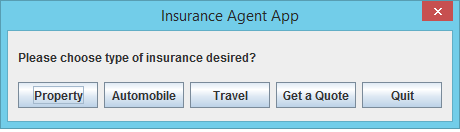
Label each file name according to the section of the assignment for which it is written. Put both sections together in a single zip file, and submit the file.

This assignment is a modification of Assignment 1. The main difference is that instead of using the **System.in.\*** and **System.out.\*** methods, your will be using the **JOptionPane** class for all the inputs / outputs.

**Section 1: Java Program File(s)**

Modify the Java program that you created from Assignment 1. According to your design, the program must:

* Use the **JOptionPane** class to create a driver program to test your Assignment 1 **InsuranceAgentApp Java** application.
* Display a new driver test program’s user interface (The interface should look like the image shown below):



* The new driver test program’s user interface from Question 1b must:
  + Include a loop to display the user interface until the user clicks the **Quit** button.
  + Instantiate a **PropertyInsurance** object and use the **JOptionPane.showInputDialog**() method to capture the **type** and **riskAmount** when the user clicks the **Property** button.
  + Instantiate an **AutomobileInsurance** object and use the **JOptionPane.showInputDialog**() method to capture the **type** and **riskAmount** when the user clicks the **Automobile** button.
  + Instantiate a **TravelInsurance** object and use the **JOptionPane.showInputDialog**() method to capture the **type** and **riskAmount** when the user clicks the **Travel** button.
  + Display the quote (**totalPremium** in the **InsuranceAgentApp** class) when the user clicks the “Get a Quote” button.
  + End when the user clicks the “**Quit**” button.

**Section 2: Screenshot of the Execution of the Implementation**

Create a screenshot of the execution and include a copy of source code of the implementation.

* Create a screenshot of the execution of the implementation of your Java program. **Note**: Go to <http://www.take-a-screenshot.org/> if you need a tutorial on taking a screenshot.
* Submit a copy of source code of the implementation.

**Section 1 and Section 2 will be graded based on the following:**

1. The program must compile, execute, produce correct results, and meet all of the specifications stated in Section 1.

Additionally you must:

1. Organize the code for user readability.
2. Organize the code for reusability.
3. Organize the code for efficiency.
4. Provide documentation with embedded comments for reader understanding.

The specific course learning outcomes associated with this assignment are:

* Compare and contrast abstract and concrete data types.
* Apply the Java concepts of file input / output.
* Design programs that handle events from user-interface components.
* Use technology and information resources to research issues in Java programing.

Grading for this assignment will be based on the following rubric.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Points: 180** | **Assignment 1: Insurance Agent App: Part 2** | | | | |
| **Criteria** | **Unacceptable**  **Below 60% F** | **Meets Minimum Expectations**  **60-69% D** | **Fair**  **70-79% C** | **Proficient**  **80-89% B** | **Exemplary**  **90-100% A** |
| 1. Specifications  Weight: 60% | The program does not compile. | The program compiles but does not execute. | The program compiles and executes but produces incorrect results. | The program compiles, executes, and produces correct results but does not meet all of the specifications. | The program compiles, executes, produces correct results, and meets all of the specifications. |
| 2. Readability Weight: 10% | The code is not organized and very difficult to read. | The code is poorly organized and difficult to read. | The code is partially organized but readable only by someone who knows the expected end result. | The code is organized and easy to read. | The code is exceptionally organized and very easy to read. |
| 3. Reusability and object-oriented programming constructs  Weight: 10% | The code is not organized for reusability. | The code is poorly organized for reusability. | The code is partially organized and some parts of the code could be reused in other programs. | The code is organized and most of the code could be reused in other programs. | The code is exceptionally organized and could be reused as a whole or each routine could be reused. |
| 4. Efficiency  Weight: 10% | The code is unnecessarily long and appears to be patched together. | The code is unnecessarily long. | The code is fairly efficient but sacrifices readability and understanding. | The code is efficient without sacrificing readability and understanding. | The code is extremely efficient without sacrificing readability and understanding. |
| 5. Documentation  Weight: 10% | No documentation is provided. | The documentation consists of embedded comments but does not help the reader understand the code. | The documentation consists of embedded comments and some header comments separating routines. | The documentation consists of embedded comments and header documentation that is useful in understanding the code. | The documentation consists of embedded comments and clearly explains what the code is accomplishing and how. |

**Weekly Course Schedule**

The purpose of the course schedule is to give you, at a glance, the required preparation, activities, and evaluation components of your course. For more information about your course, whether on-ground or online, access your online course shell.

The expectations for a 4.5 credit hour course are for students to spend 13.5 hours in weekly work. This time estimate includes preparation, activities, and evaluation regardless of the delivery mode.

**Instructional Materials**

In order to be fully prepared, obtain a copy of the required textbooks and other instructional materials prior to the first day of class. When available, Strayer University provides a link to the first three (3) chapters of your textbook(s) in eBook format. Check your online course shell for availability.

Review the online course shell or check with your professor to determine whether Internet-based assignments and activities are used in this course.

Strayer students are encouraged to purchase their course materials through the Strayer Bookstore. [http://www.strayerbookstore.com](http://www.strayerbookstore.com/).  If a lab is required for the course, the Strayer Bookstore is the only vendor that sells the correct registration code so that Strayer students may access labs successfully.

**Software**

The Java Platform and an Integrated Development Environment (IDE) is required for this course.

* Download the Java Development Kit (JDK) from <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
* Download the Eclipse IDE from <http://www.eclipse.org/downloads/>. Most users will want the Eclipse IDE for Java Developers, Windows 32 bit.

Students may use other Java IDEs to develop the code for their assignments and labs as long as the code and assignments can be replicated for the instructor’s review. A suggested alternative to Eclipse that some consider less powerful but easier to use is JGrasp. JGrasp can be downloaded for Windows and Mac from <http://spider.eng.auburn.edu/user-cgi/grasp/grasp.pl?;dl=download_jgrasp.html>.

**Kaltura Video Sessions**

Instructors will record video or screen capture sessions using **Kaltura** in Weeks 2 and 4. Your instructor may also provide these sessions on a weekly basis to act as student tutorials on the following applicable procedures:

* Running and navigating the software or labs that will be used in this course
* Demonstrating the use of relevant programming languages and tools through screen capture and live navigation
* Logging into the lab environments and running a sample lab required in this course (if labs are required)
* Demonstrating the downloading and installing of software to a student’s computer (if necessary) or using the software already installed in the Strayer University campus labs
* Locating or finding files and other student materials that may be required to use in assignments
* Submitting assignments in Blackboard

You are strongly advised to use the material provided by faculty in these sessions. Online and Ground instructors will post the Kaltura video recorded sessions to the Instructor Insights folder within the weekly tabs of the online course shell (Blackboard).

**Discussions**

To earn full credit in an online threaded discussion, students must have one original post and a minimum of one other post per discussion thread.

Please note: Material in the online class will be made available three weeks at a time to allow students to work ahead, however, faculty will be focused on and responding only to the current calendar week. As it is always possible that students could lose their work due to unforeseen circumstances, it is a best practice to routinely save a working draft in a separate file before posting in the course discussion area.  
  
Professors hold discussions during class time for on-ground students. Check with your professor if any additional discussion participation is required in the online course shell outside of class hours.

**Tests**

Tests (quizzes, midterm and final exams, essay exams, lab tests, etc.) are available for student access and completion through the online course shell. Check the online course shell to determine how students are expected to take the tests. Do not change these questions or their point values in any way. This disrupts the automated grade book preset in the online course shell.

* Online students are to complete the test by Monday 9:00 a.m. Details regarding due dates are posted in the Blackboard Calendar tool.
* On-ground students are to complete the tests after the material is covered and before the next class session.

**Assignments**

A standardized performance grading rubric is a tool your professor will use to evaluate your written assignments. Review the rubric before submitting assignments that have grading rubrics associated with them to ensure you have met the performance criteria stated on the rubric.

Grades are based on individual effort. There is no group grading; however, working in groups in the online or on-ground classroom is acceptable.

Assignments for online students are always submitted through the online course shell. On-ground professors will inform students on how to submit assignments, whether in paper format or through the online course shell.

**Association for Computing Machinery (ACM) Digital Library**

The ACM Digital Library is a complete collection of all of ACM’s publications, including ACM journals, conference proceedings, magazines, newsletters, and multimedia titles. The ACM Digital Library contains the largest and most complete full-text archive of articles on computing available today, consisting of: 2.0+ million pages of full-text articles, 20,000 new full-text articles added each year, 40+ high-impact journals, 270+ conference proceedings titles, 9 magazines (including the flagship Communications of the ACM), and 43 special interest groups contributing content.

You are encouraged to search the ACM Digital Library for full-text articles for your writing assignments and term papers refereed referenced material. For more information on the ACM Digital Library, please watch the video located in the Student Center tab of the online course shell (Blackboard).

To access the ACM Digital Library:

Students:

1. Login to iCampus: <http://icampus.strayer.edu>
2. Click on Campus & Library
3. Mouse-over or click on Learning Resource Center
4. Click on Databases
5. Scroll down to Information Systems / Computing
6. Click on ACM Digital Library

**Resources**

The Resource Center navigation button in the online course shell contains helpful links. Strayer University Library Resources are available here as well as other important information. You should review this area to find resources and answers to common questions.

Technical support is available for the following:

* For **technical questions**, please contact Strayer Online Technical Support by logging in to your iCampus account at [https://icampus.strayer.edu/login](https://icampus.strayer.edu/login%20) and submitting a case under “Student Center,” then “Submit Help Ticket.” If you are unable to log in to your iCampus account, please contact Technical Support via phone at (877) 642-2999.
* For **concerns with your class**, please access the Solution Center by logging in to your iCampus account at [https://icampus.strayer.edu/login](https://icampus.strayer.edu/login%20) and submitting a case under “Student Center,” then “Submit Help Ticket.” If you are unable to log in to your iCampus account, please contact the IT Help Desk at (866) 610-8123 or at <mailto:IThelpdesk@Strayer.edu>.

TurnItIn.com is an optional online tool to assess the originality of student written work. Check with your professor for access and use instructions.

The **Strayer Policies** link on the navigation bar in the online course shell contains academic policies. It is important that students be aware of these policies.