# Computer Science 1

## "Programming in Alice"
## Topic Outline
## Fort Atkinson High School
## Instructor – Dean Johnson

Unit 1 – Getting Started with Alice

Unit 2 – Program Design and Implementation

Unit 3 – Programming: Putting Together the Pieces

Unit 4 – Classes, Objects, Methods and Parameters

Unit 5 – Interaction: Events and Event Handling

Unit 6 – Functions and If/Else

Unit 7 – Repetition: Definite and Conditional Loops

Unit 8 – Lists and List Processing

Unit 9 – Variables and Arrays

Unit 10 - Final Project

# Grading Rubrics

| Grading Criteria | Percent of Grade |
|---|---|
| Problem Sets | 20% |
| Tests | 20% |
| Run | 20% |
| Code | 10% |
| Effort | 10% |
| On Time | 10% |
| Storyboard | 10% |

| | Run: The visual running of the program |
|---|---|
| 4 | **Excellent Work.** The scenario is played out correctly, accurately and additional creative effort was put into making the scenario unique. |
| 3 | **Good work.** The scenario is played out correctly and accurately. The expectations with objects, methods, parameters and variables are met. |
| 2 | **Fair work.** The scenario is missing one component and therefore, does not execute as outlined. |
| 1 | **Poor work.** The scenario is missing one or more components and therefore, does not execute as outline. Major pieces of the scenario are either not completed or not addressed. |
| 0 | **No credit.** The running of the program produces nothing like the expected scenario. |

| | Code: The hard-copy version of the program |
|---|---|
| 4 | **Excellent Work.** The printed code is complete. Very descriptive comments describe each major section of the program. Methods are commented in a very descriptive manner. Code is well-written and elegant. Appropriate names are chosen for all methods, parameters, and variables. |
| 3 | **Good work.** The printed code is complete. Comments are included for all major sections of the program. All methods have a comment block at the beginning of the method that describes the purpose of the method. Code is written in a logical piece-wise fashion and not in a "Rube-Goldberg" fashion. All appropriate parameters and variables are included. |
| 2 | **Fair work.** Code is not complete. Maybe some comments are missing that could better explain the direction of the code. Code may be not well written and is hard to follow. Not all required parameters or variables are included. |
| 1 | **Poor work.** Code is not complete. Program is not commented. Code is poorly written. Required parameters or variables are missing. |
| 0 | **No credit.** The code is either not turned in or does not represent the problem. |

| | Storyboard: The plan for the program |
|---|---|
| 4 | **Excellent Work.** All descriptions of scenes are very complete. Diagrams clearly explain what actions are taking place in each scene. All pertinent objects, methods, and variables are clearly identified. |
| 3 | **Good work.** All descriptions of scenes are complete. Diagrams explain what actions are taking place. All objects, methods and variables are identified. |
| 2 | **Fair work.** Descriptions of scenes are incomplete. Diagrams do not explain well the actions that are taking place. Some objects, methods or variables are missing |
| 1 | **Poor work**. Descriptions of scenes are very incomplete. Diagrams are either missing or poorly explain the actions that are taking place. Many objects, methods, or variables are missing. |
| 0 | **No credit.** Storyboard was not turned in. |

| | Effort: Daily work ethic |
|---|---|
| 4 | **Excellent Work.** Student is on task each day. Pays attention to lessons and makes daily progress on the current project. Student is cooperative and willing to help other students as needed to help solve a problem. |
| 3 | **Good work.** Student is on task each day. Pays attention to lessons and makes daily progress on the current project. |
| 2 | **Fair work.** Student is not on task each day. Student had to be re-directed by the teacher at some point. Daily progress on current project is slow. |
| 1 | **Poor work.** Student is often off task. Student is talking or not paying attention during lessons. Student often needs to be re-directed by the teacher to remain on task. Daily progress on current project is very slow. |
| 0 | **No credit.** Student daily work is way below expectations. |

| | On-time: Respect for a deadline |
|---|---|
| 4 | **Excellent Work.** The program was visually checked on the due date. |
| 3 | **Good work.** The program was visually checked one day late. |
| 2 | **Fair work.** The program was visually checked two days late. |
| 1 | **Poor work.** The program was visually checked three days late. |
| 0 | **No credit.** The program was more than three days late. |

# Unit 1 – Getting Started with Alice

1. **Vocabulary to learn**
   - Computer Program
   - Program Design
   - Flowchart
   - Class
   - Object
   - Three Dimensional
   - 3D Model
   - Virtual World
   - Animation
   - Orientation

2. **Problem Set #1: (15 points)**
   **These questions MUST be answered carefully and completely to receive full credit. (See Problem Set Worksheet)**
   1. What are the six directions an Alice object can move?
   2. What does it mean to say that an object is egocentric in terms of motion?
   3. Is the center of an Alice object always located at its center of mass? Justify your answer.
   4. What does the phrase "Add instance to World" mean?
   5. Explain what a computer program is.

3. **Required Project: (12 points)**

   **"My First Scene" (Rubrics: Run, On-time, Effort)**

   1. If you haven't already done so, create a folder on your H: drive called ComputerScience1. In that folder, create another folder called Alice Projects. This is where you will save all of your Alice Projects.
   2. Create a new World.
   3. Choose the "Grass" template.
   4. Save your world as ***FirstNameLastNameProjectTitle*** (Where the last three letters are your first, middle, and last initials.)
   5. Choose 3 animals objects from the Animal Folder in the local gallery.
   6. Choose 3 plant objects from the Nature Folder in the local gallery.
   7. Choose 1 person object from the People Folder in the local gallery.
   8. Use the methods for each of the animal objects so that each is facing the person object.
   9. Arrange your world so that all 7 objects can be seen.
   10. Press PLAY to test your world
   11. Take a Picture of your world while your world is running. Pay attention to where this picture is being saved and save it on your H: drive. Be to save the file in the following format: F*irstnameLastnameMyPicture.*
       *Example: DeanJohnsonMyPicture*
   12. Email this picture to me at djohnson@mail.fortschools.org
   13. You are done with your first Alice World!

Name _____ Hour ___

# Computer Science 1
# Programming in Alice
# Problem Set #1 (15 points)

**These questions MUST be answered carefully and completely to receive full credit.**

1. What are the six directions an Alice object can move?

2. What does it mean to say that an object is egocentric in terms of motion?

3. Is the center of an Alice object always located at its center of mass? Justify your answer.

4. What does the phrase "Add instance to World" mean?

5. Explain what a computer program is.

# Unit 2 – Program Design and Implementation

1. **Vocabulary to learn**
   - Algorithm
   - Bug
   - Comment
   - Implement
   - Instruction
   - Method
   - Property of an Object
   - Runtime
   - Scenario
   - State of an object
   - Storyboard
   - Nested
   - Syntax
   - Trial-and-Error

2. **Problem Set #2 (15 points)**
   1. Creating a computer program can be described as a four-step process using problem solving techniques. List the four steps and give a brief description of each step.
   2. *Do together* and *Do in order* are each described as "control statements." Why?
   3. What is meant by the phrase "one block of code may be nested in another"?
   4. How do you know if a program has a bug? Explain.
   5. Since comments are ignored by Alice, why do we add comments to our programs?

3. **Required Project: (16 points)**

   **"SnowPeople" (Rubrics: Run, Code, On-time, Effort)**

   1. Create a program that carries out the scene described below.
   2. Save this program as ***FirstNameLastNameSnowPeople.***
   3. Place comments in your program to describe essential blocks of code.
   4. Export the code for printing and print your code

   ***Scenario:***
   A snowman is trying to meet a snowwoman who is talking with a friend (another snowwoman.) The snowman tries to get her attention. He turns to face the snowwoman and says "Hello!" She turns to look at the snowman and he blinks his eyes at her. She blushes (her head turns red.) But, alas, she is not interested in meeting him. She gives him a "cold shoulder" and turns back to talk with her friend. He disappointingly hangs his head and turns away.

Name _____ Hour ___

# Computer Science 1
# Programming in Alice
# Problem Set #2 (15 points)

**These questions MUST be answered carefully and completely to receive full credit.**

1. Creating a computer program can be described as a four-step process using problem solving techniques. List the four steps and give a brief description of each step.

2. *Do together* and *Do in order* are each described as "control statements." Why?

3. What is meant by the phrase "one block of code may be nested in another"?

4. How do you know if a program has a bug? Explain.

5. Since comments are ignored by Alice, why do we add comments to our programs?

# Unit 3 – Programming
# Putting Together the Pieces

1. **Vocabulary to learn**
   - Boolean Value
   - Condition
   - Conditional Execution
   - Control Structure
   - Count
   - Decision
   - Expression
   - Function
   - Instruction
   - Loop
   - Relational Operator
   - Repetition

2. **Problem Set #3 (15 points)**

   1. Explain the difference between a method and a function.
   2. What are the four operations that can be used in a math expression in Alice?
   3. What is the purpose of a Loop statement?
   4. Explain what a condition is and how conditionals are used in computer programming.
   5. What are the six relational operators in Alice and explain the meaning of each.

3. **Required Project: (20 points)**

   **"SkatePark" ( Rubrics: Run, Code, On-time, Storyboard, Effort)**

   ***Scenario:***
   Create a skate park. A girl skateboarder rides her board, does a 360, jumps over a box and does one other trick. Another person watching the girl comments on her board skills.

   1. Create a storyboard for the scenario.
   2. Create a program that carries out the scene described below.
   3. Save this program as ***FirstNameLastNameSkatePark.***
   4. Place comments in your program to describe essential blocks of code.
   5. Export the code for printing and print your code.
   6. I will do a visual check of your program by watching a running of it.

Name _____ Hour ___

# Computer Science 1
# Programming in Alice
# Problem Set #3 (15 points)

**These questions MUST be answered carefully and completely to receive full credit.**

1. Explain the difference between a method and a function.

2. What are the four operations that can be used in a math expression in Alice?

3. What is the purpose of a Loop statement?

4. Explain what a condition is and how conditionals are used in computer programming.

5. What are the six relational operators in Alice and explain the meaning of each.

# Unit 4 – Interaction: Events and Event Handling

1. **Vocabulary to learn**
   - Control of flow
   - Event
   - Event Handling Method
   - Incremental Development
   - Interactive

2. **Problem Set #4 (15 points)**

   1. In what ways is an interactive animation application different from a non-interactive animation application?
   2. How does an event work?
   3. What would be the purpose in including a parameter in an event handling method?
   4. What is incremental development and why is it a good technique when writing event-driven applications?
   5. Name three applications that you use that are event driven. Give an example for each as to why these applications are considered to be event-driven.

3. **Required Project (16 Points) ( Rubrics: Run, Code, On-time, Effort)**

   **Option 1: "Turtle Motion Controller"**

   ***Scenario:***
   In this project, you are to create a turtle motion controller to help a turtle perform exercises for his upcoming race with the rabbit. Create a world that contains a turtle and then create the following motion control methods for the turtle:

   - **headBob:** allows the turtle's head to bob a little
   - **tailWag:** allows the turtle's tail to wag
   - **oneStep:** allows the turtle to move forward one step: his legs should move while he is taking that one step
   - **walkForward**: combines the above three methods, to make a realistic step; all movements should take the same amount of time and occur at the same time
   - **turnAround:** turns the turtle 180 degrees; he should be walking while turning
   - **turnLeft**: turns the turtle left, walking while he is turning
   - **turnRight:** turns the turtle right, walking while he is turning
   - **hide:** allows the turtle to hide in his shell (you may assume that the turtle is currently outside of his shell); remember not to leave the shell hanging in midair

- **reappear:** allows the turtle to reappear from his shell (you may assume that the turtle is currently hidden)
- **talk:** has the turtle look at the camera and say something to the user.

**Create the following keyboard controls:**

- When the up arrow key is pressed, the turtle is to walk forward.
- When the down arrow key is pressed, the turtle is to turn around (180 degrees)
- When the left arrow key is pressed, the turtle is to turn left.
- When the right arrow key is pressed, the turtle is to turn right.
- When the letter "H" is pressed, the turtle is to hide in his shell.
- When the letter "R" is pressed, the turtle is to reappear from his shell.
- When the letter "T" is pressed, the turtle is to talk to the user.

**Details:**

1. Create a program that carries out the scenario described below.
2. You must include *Class methods* as described above.
3. You must include *Events* as described above.
4. *Rename* your turtle object.
5. Save this program as ***FirstNameLastNameTurtle.***
6. Each method should start with a comment that describes exactly what the method does.
7. Export the code for printing and print your code.
8. I will do a visual check of your program by watching a running of it.

## 4. Project (Option 2)

### "NFL Robot Motion Controller"

***Scenario:***

In this project, you are to create an NFL Robot motion controller. You will be able to control the Robot by creating *events* that react to *key* commands. Create a world that contains a Robot and then create the following motion control methods for the robot:

***Methods:***

- **headBob:** allows the robot's head to bob forward just a little, then back to normal position
- **armWag:** allows the robot's arms to wag slightly forward, then back, and then back to original position. The arms should wag alternately and simultaneously.
- **oneStep:** allows the robot to move forward one step with both legs. His leg joints should move while he is taking that one step.

- **walkForward**: combines the above three methods, to make a realistic step; all movements should take the same amount of time and occur at the same time
- **turnAround:** turns the robot 180 degrees; he should be walking while turning
- **turnLeft**: turns the robot left, walking while he is turning
- **turnRight:** turns the robot right, walking while he is turning
- **hide:** makes the robot disappear
- **reappear:** allows the robot to reappear
- **armSwing:** swings both left and right arms in front of robot
- **jumpUpAndDown:** the robot jumps up and down
- **headRotate:** the robots head turns left, right, forward, back, then side to side.
- **talk:** has the robot look at the camera and say something to the user.
- **"One other Action"** – This action is designed by you. The action *must* return the robot to its original position.

**Create the following keyboard controls:**

- When the up arrow key is pressed, the robot is to walk forward.
- When the down arrow key is pressed, the robot is to turn around (180 degrees)
- When the left arrow key is pressed, the robot is to turn left.
- When the right arrow key is pressed, the robot is to turn right.
- When the letter "H" is pressed, the robot is to hide.
- When the letter "R" is pressed, the robot is to reappear.
- When the letter "T" is pressed, the robot is to talk to the user.
- Make events for each of the methods for your robot

**Details:**

1. Create a program that carries out the scenario described below.
2. You must include *Class methods* as described above.
3. You must include *Events* as described above.
4. *Rename* your robot object.
5. Save this program as ***FirstNameLastNameRobot.***
6. Each method should start with a comment that describes exactly what the method does.
7. Export the code for printing and print your code.
8. I will do a visual check of your program by watching a running of it.

# Computer Science 1
# Programming in Alice
# Problem Set #4 (15 points)

Name _____

Hour ___

**All questions MUST be answered fully and completely to receive full credit.**

1. In what ways is an interactive animation application different from a non-interactive animation application?

2. What does it mean to have a method called when an event is handled? Explain.

3. Are event-driven programs executed sequentially? Explain.

4. What is incremental development and why is it a good technique when writing event-driven applications?

5. In the real world, there are many examples of event-driven applications. Name three applications that you use that are event driven. Give an example for each as to why these applications are considered to be event-driven.

# Unit 5 – Classes, Objects, Methods and Parameters

1. **Vocabulary to learn**
   - Abstraction
   - Argument
   - Calling a Method
   - Class
   - Class-level method
   - Inheritance
   - Instance
   - Instantiate
   - Method
   - Object
   - Parameter
   - Primitive Method
   - Stepwise refinement
   - World-level Method
   - IsShowing
   - Opacity

2. **Problem Set #5 (15 points)**
   1. Is it possible to have more than one instance of the same class in a program? Explain.
   2. When you run your program, what method is called (give the complete name)?
   3. Explain what is meant by the term "class-level" method?
   4. What is meant by the term parameter? How are parameters used when invoking methods?
   5. What does it mean if a variable or parameter is declared as a Boolean type?

3. **Required Project 5: (20 points)**

   **"Under the Big Top" (Rubrics: Run, Code, On-time, Storyboard, Effort)**

   ***Scenario:***
   A boyfriend and girlfriend go to the Circus. There are rides, side-shows and arcade games at the circus. The boyfriend plays a game in order to win his girlfriend a prize. Another boy, who is admiring the girlfriend, tries to beat the boyfriend at the game but fails in his attempt. The girlfriend and boyfriend go on a ride and leave with her prize.

   1. Create a storyboard for the scenario.
   2. Create a program that carries out the scene described below.
   3. You must include *dialog* in the program.
   4. You must include *sound* in the program.
   5. You must include at least one *Class method*.
   6. You must include a parameter in at least one class method.
   7. Save this program as ***FirstNameLastNameBigTop.***
   8. Place comments in your program to describe essential blocks of code.
   9. Export the code for printing and print your code.
   10. I will do a visual check of your program by watching a running of it.

# Computer Science 1
# Programming in Alice
# Problem Set #5 (15 points)

Name _____

Hour ___

**These questions MUST be answered carefully and completely to receive full credit.**

1. Is it possible to have more than one instance of the same class in a program? Explain.

2. When you run your program, what method is called (give the complete name)?

3. Explain what is meant by the term "class-level" method?

4. What is meant by the term parameter? How are parameters used when invoking methods?

5. What does it mean if a variable or parameter is declared as a Boolean type?

# Unit 6 – Functions and If/Else

1. **Vocabulary to learn**
   - Boolean Expression
   - Conditional Execution
   - Conditional Expression
   - Variable
   - Expression
   - Function
   - If/Else Statement
   - Integer
   - Logical Operator
   - Random Number
   - Relational Operator
   - Return Statement

2. **Problem Set #6 (15 points)**
   1. A conditional expression in an *If/Else* statement **must** evaluate to what type of value? Why?
   2. Explain a situation where a programmer would choose to use the built-in random generator.
   3. Why must a function have a *Return* statement?
   4. Can a function have more than one Return statement? Explain.
   5. If a Boolean expression needs to have more than one condition, what two operators may be used to connect the conditions?

3. **Required Project: (20 Points)**

   ***"ClickerGame" (*** **Rubrics: Run, Code, On-time, Storyboard, Effort)**

   ***Scenario:*** You are to create a user-interactive game in which the user earns points by clicking on objects that randomly appear on the screen. The game ***must*** have the following components.
   - An Instruction Screen
   - A Start Button
   - A Score that is kept while the user is playing
   - An Object that, when pressed, earns the user points
   - An Object that, when pressed, takes points away from the user
   - A Winning Score that, if reached, allows the user to win the game
   - A Winner or Loser screen that reports to the user when they finish playing the game

**Details:**
1. Create a storyboard for the scenario.
2. You must consider the best use of methods and events in your design.
3. Create a program that carries out the scene described above.
4. You must use the *random* generator in the program.
5. You must include *sound* in the program.
6. Save this program as ***FirstNameLastNameClickerGame.***
7. Place comments in your program to describe essential blocks of code.
8. Export the code for printing and print your code.
9. I will do a visual check of your program by watching a running of it.

# Computer Science 1
# Programming in Alice
# Problem Set #6 (15 points)

Name _____

Hour ___

**All questions MUST be answered fully and completely to receive full credit.**

1. A conditional expression in an *If/Else* statement **must** evaluate to what type of value? Why?

2. Explain a situation where a programmer would choose to use the built-in random generator.

3. Why must a function have a *Return* statement?

4. Can a function have more than one Return statement? Explain your answer.

5. If a Boolean expression needs to have more than one condition, what two operators may be used to connect the conditions?

# Unit 7 – Repetition: Definite and Conditional Loops

1. **Vocabulary to learn**
    1. Count
    2. Definite Loop
    3. Indefinite Loop
    4. Infinite Loop
    5. Nested Loops
    6. While

# Unit 8 – Lists and List Processing

1. **Vocabulary to learn**
    1. Data Structure
    2. Iterate through a List
    3. List
    4. List Search
    5. List Variable
    6. For all in Order
    7. For all Together

# Unit 9 – Variables and Arrays

1. **Vocabulary to learn**
    1. Access
    2. Mutable Variable
    3. State of an Object
    4. Array
    5. Property of an Object
    6. Index Variable
    7. Size of an Array
    8. Array Visualization

# Unit 10 - Final Project

Alice – Final Project – Peer Critique

Name of Project: _____

| Description | Rating | | | | | |
|---|---|---|---|---|---|---|
| | 5 | 4 | 3 | 2 | 1 | 0 |
| | Awesome! | | Ok | | Not very good | Not Applicable |
| Instruction Screen | | | | | | |
| Ease of Use | | | | | | |
| Entertainment Value (Fun) | | | | | | |
| Graphics | | | | | | |
| Sound | | | | | | |
| Challenge | | | | | | |
| Originality | | | | | | |
| **DO NOT CALCULATE A TOTAL. INSTEAD, GIVE AN OVERALL SCORE.** | | | | | | |
| **OVERALL SCORE** | | | | | | |

What did you like best about this project?

What did you like least about this project?

What would you change in the project to make it better?

Acknowledgements:

"Learning to Program with Alice". Dann, Cooper, Pausch