Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A31

Game C/S Model – Collaboration Diagram

Team:

Akpoguma Daniella - Id: 041075624 / Philogene Villanueva - Id: 041063813

TM Model C/S Proposal

***This template is suggested (not mandatory) to answer A31 Specification.***

|  |  |
| --- | --- |
| **Part**  **1** | **C/S Architecture** |

* 1. **UC Model (1pt)**

🛠 Define the UC diagram and the UC table (actors and functionalities).

**A screenshot of a computer screen

Description automatically generated**

**Fig. 1** – C/S Model for Chat[[1]](#footnote-1)

**UC Diagram** (change this diagram to accommodate the actors and functionalities to be used):

**Actors table** (example):

|  |  |
| --- | --- |
| **Actors** |  |
| [Actor name] | [Description] |
| Server | Represents the server side component |
| User | Represents the person using the application |
|  |  |

**UC table** (example):

|  |  |
| --- | --- |
| **Use Cases** | **USER** |
| [UC Name] | [Description] |
| Login | Allows user to login the Turing Machine Application |
| View Task | Enables the user to view the list of task |
| Create Task | Allows user to create task |
| Update Task Status | Enables user to update the status of a task |
| Sync Task | Synchronizes task between user and server |
|  |  |

|  |  |
| --- | --- |
| **Use Cases** | **SERVER** |
| [UC Name] | [Description] |
| Authenticate User | Authenticates user during login process |
| Fetch Task | Retrieves task from server to display to the user |
| Update task Status | Allows server to update the status of a task |
| Synch Task | Synchronizes task between user and server |

* 1. **ClassD (2pt)**

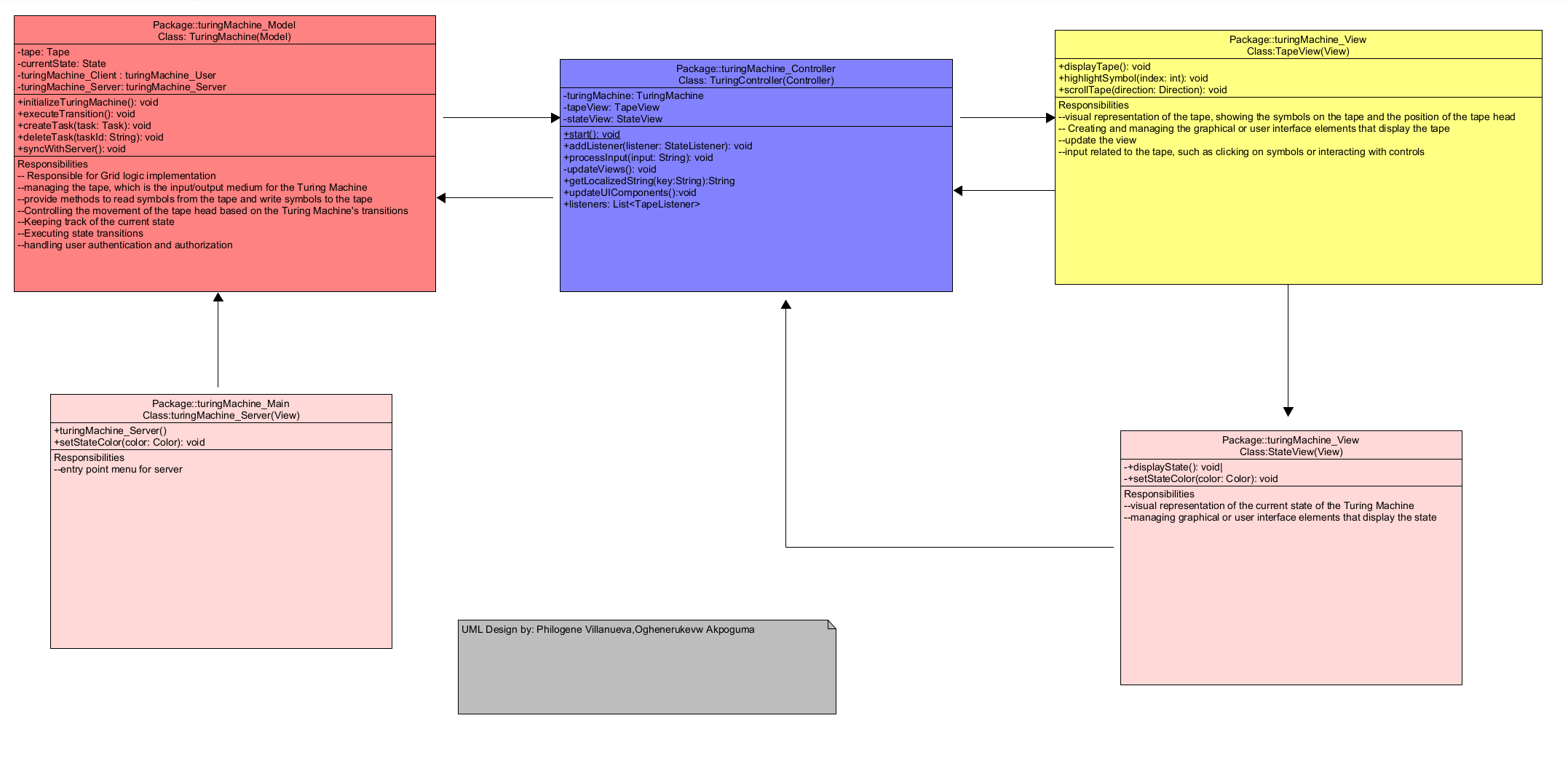
🛠 To draw the diagram, you can use tools (ex: <https://app.diagrams.net/>) or desktop applications (ex: Visio / Powerpoint) or simply take photos from drawings.

**Class Diagram** (change this diagram to accommodate the actors and functionalities to be used):

A diagram of a computer program

Description automatically generated with medium confidence

UML for User



UML for Server

**Fig. 3** – ClassD for a chat[[2]](#footnote-2)

**Class table** (example):

|  |  |
| --- | --- |
| **Class name** | **TuringMachine (Model)** |
| Inner Fields[[3]](#footnote-3) | tape: instance of tapeView  currentState: instance of the current state  user: User instance  server: Server instance |
| Relationships[[4]](#footnote-4) | TuringController class |
| Methods | initializeTuringMachine(): void  executeTransition(): void  createTask(task: Task): void  deleteTask(taskId: String): void  syncWithServer(): void |

|  |  |
| --- | --- |
| **Class name** | **TuringController (Controller)** |
| Inner Fields[[5]](#footnote-5) | turingMachine: instance of model class  tapeView : instance of tapeView  stateView: instance of stateView |
| Relationships[[6]](#footnote-6) | TuringController class  TapeView class  StateView class |
| Methods | initializeTuringMachine(): void  executeTransition(): void  createTask(task: Task): void  deleteTask(taskId: String): void  syncWithServer(): void |

|  |  |
| --- | --- |
| **Class name** | **TapeView (View)** |
| Inner Fields[[7]](#footnote-7) |  |
| Relationships[[8]](#footnote-8) | TuringController class  StateView class |
| Methods | displayTape(): void  highlightSymbol(index: int): void  scrollTape(direction: Direction): void |

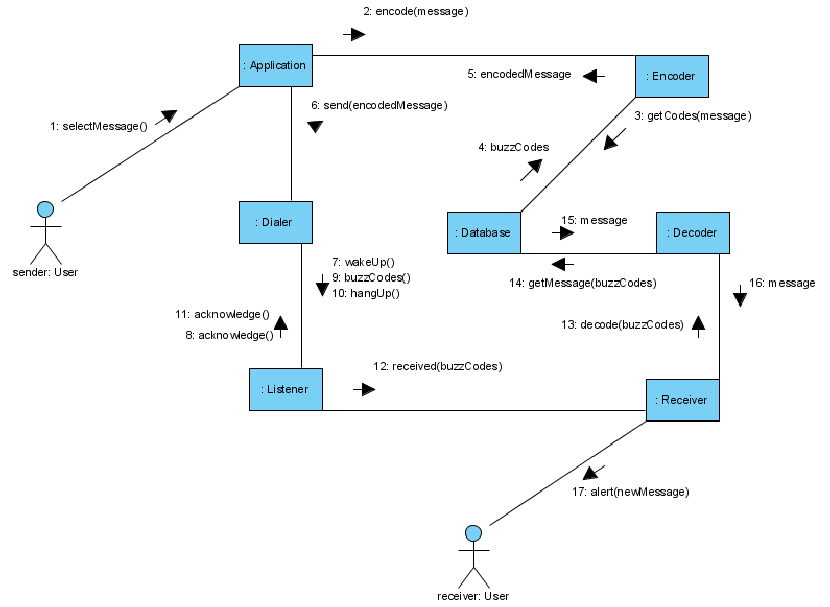
|  |  |
| --- | --- |
| **Class name** | **StateView (View)** |
| Inner Fields[[9]](#footnote-9) |  |
| Relationships[[10]](#footnote-10) | TuringController class  TapeView class |
| Methods | displayState(): void|  setStateColor(color: Color): void |

*Create tables for all classes.*

* 1. **UML Collaboration Diagram**

*Describe the collaboration diagram to be used.*

* + - ***Example****:*



**Fig. 4** – Collab model for a chat[[11]](#footnote-11)

**References**

*[Include eventual references used here]*

Algonquin College

Fall, 2023

1. See <https://www.researchgate.net/figure/System-Architecture-Use-Case-Diagram-Client-side-Functionality_fig2_318502492>. [↑](#footnote-ref-1)
2. See <https://www.codeproject.com/Articles/443660/Building-a-basic-HTML-client-server-application>. [↑](#footnote-ref-2)
3. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-3)
4. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-4)
5. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-5)
6. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-6)
7. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-7)
8. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-8)
9. The inner fields and relationships together are properties from the class. [↑](#footnote-ref-9)
10. In the diagram, you can see relationships (ex: association / aggregation) between classes. In the implementation, these relations imply in the inclusion of other classes as fields. [↑](#footnote-ref-10)
11. See <https://www.researchgate.net/figure/Collaboration-diagram_fig1_268362338>. [↑](#footnote-ref-11)