University of Cape Town

CSC2002S

Computer Science

Assignment 02 Report

Author: Nkosinathi Ntuli	*Student Number:* NTLNKO007

August 28, 2022

a) A description of the HungryWordMover class.

The HungryWordMover class facilitates the movement of the hungry word, it has the same attributes and methods as the WordMover class but instead of using the dropped() method it uses drifted() since the hungry word drifts across the screen and it uses drift() instead of drop() and additional pause() method was added to pause moving the hungry word when needed.

b) A description of any other classes you added (and a justification of why they were necessary).

No additional classes were added as they were not necessary for the implementation of the code, everything was easily done by updating the given code.

c) A broad description of modifications you made to the existing classes and how you ensured thread safety.

1. CatchWord class
   The first modification was for catching the lowest word on the panel if there were duplicate falling words. In the run method I added a second while loop after the first to find if there are words that matches (a duplicate) the typed word and check the lowest, once done reset it. The match method was modified to not reset the word after matching and the reset was done in the CatchWord class instead.
2. FallingWord class
   The class constructor was extended to include the y coordinate, maximum x coordinate and a boolean to keep track of the hungry word. The setX() method was modified to check if the x coordinate does not exceed the set maximum x coordinate.The resetPos() modified to check whether a word is a normal or a hungry word since they move in different axes. Added a drop() method to immediately drop a word when it bounces on the hungry word. dift(), drifted() were added as mentioned in question a). Lastly isHungry() to check if the current word is a hungry word.
3. GamePanel class
   In the paintComponent() method I modified the code so that the hungry word is coloured green. Added a collide() method that checks for each normal word if it collides with the current hungry word
4. TypingTutorApp class
   Modified it so that it creates the hungry word using the last word on the list of FallingWord[] and creates a HungryWordMover to move it.

d) Any race conditions that you identified in the original code (for extra credit).

The code can be compiled by running **make** on the command line in the root folder (i.e folder containing the src, bin folders and the Makefile file).
To run the code **make run** is used, if you want to provide the input to TypingTutorApp run **make run input=”**<total no of words> <no of words on falling at point> <text file name that contains a list of words to use (optional)>**”**