Pseudocode for Assignment 4 – Fit Gym

Santiago Velasquez Talbott [ID 40321098]

Algorithm for Class GymPasses

1. Declare five private integer variables to keep track of how many Regular, Student, Senior, Weekend, and Weekly passes have been sold
2. Declare public static final constants that hold the fixed prices: Regular $7, Student $5, Senior $4, Weekend $12, Weekly $42
3. Provide a default
4. Initialize all five counts to zero so the object starts empty
5. Provide a parameterized constructor that receives five integers
6. Store each incoming integer into its corresponding private attribute
7. Provide a copy constructor that receives another GymPasses object
8. Copy the five counts from the given object into the new one
9. Provide simple accessor methods for each of the five counts
10. Each method simply returns the value of its attribute
11. Provide simple mutator methods for each count
12. Each method updates its attribute with the new value passed as parameter
13. Provide a method addGymPasses that receives five integers
14. Add each received number to the corresponding existing count
15. Provide a method gymPassesTotal with no parameters
16. Calculate the total monetary value by multiplying each count by its price and summing everything
17. Return that total amount
18. Provide a toString method
19. Return a string that clearly shows the number of passes of each type along with their individual price, in the exact format required by the assignment
20. Provide an equals method that receives an Object
21. If the object is not a GymPasses, return false
22. Otherwise compare all five counts and return true only if they are exactly the same

Algorithm for Class GymCard

1. Declare private attributes: type and name as String, day and month as integers
2. Provide a default constructor

3. Leave all fields with their default values (null for Strings, 0 for integers)
4. Provide a constructor that receives type, name, day, and month
5. Store the type and name directly
6. If the day is between 1 and 31 inclusive, store it; otherwise store 0
7. If the month is between 1 and 12 inclusive, store it; otherwise store 0
8. Provide a copy constructor that receives another GymCard
9. Copy all four attributes from the given card into the new one
10. Provide accessor methods for type, name, day, and month
11. Each returns its corresponding field
12. Provide a setDay method
13. Validate the day (1–31); if valid store it, otherwise store 0
14. Provide a setMonth method
15. Validate the month (1–12); if valid store it, otherwise store 0
16. Provide a toString method
17. Format the day and month with a leading zero when necessary
18. Return exactly three lines: “Gym Card: …”, “Holder: …”, “Expires: dd/mm”
19. Provide an equals method that receives an Object
20. Return true only if type, name, day, and month are exactly identical to the other card

Algorithm for Class Reception

1. Declare two private attributes: one GymPasses object and one array of GymCard objects (the array may be null)
2. Provide a default constructor
3. Create a new GymPasses object containing zero passes
4. Set the card array to null (meaning no membership cards yet)
5. Provide a constructor that receives a GymPasses object and a GymCard array
6. Make a deep copy of the given GymPasses object (so changes won’t affect the original)
7. If the card array is null, keep it null
8. Otherwise create a brand-new array and copy each GymCard using its copy constructor
9. Provide isEqualValue method that receives another Reception
10. Return true if both receptions have exactly the same total dollar value of gym passes
11. Provide isEqualAmount method that receives another Reception
12. Return true if both receptions have exactly the same number of each of the five pass types
13. Provide totalValue method
14. Simply return the total monetary value of the passes (by calling the GymPasses method)
15. Provide totalGymCards method
16. If the card array is null return 0, otherwise return the current length of the array
17. Provide addGymCard method that receives a new GymCard

18. Create a larger array (or a size-1 array if currently null)
19. Copy all existing cards into the new array and place the new card at the end
20. Provide removeGymCard method that receives an index
21. If the index is invalid or there are no cards, return false
22. Create a smaller array, copy all cards except the one at the given index
23. If it was the last card, set the array to null
24. Return true when removal is successful
25. Provide updateGymCardExpiry method that receives index, new day, and new month
26. If the index is valid, use the card's setters to update day and month
27. Provide addGymPasses method that receives five integers
28. Forward the five numbers to the GymPasses object and return the new total dollar value
29. Provide equals method that receives an Object
30. Return true only if the total dollar value of passes is the same AND the number of membership cards is the same
31. Provide toString method
32. Show the full breakdown of passes
33. Then either display each membership card (using its toString) or display "No membership card"

Algorithm for Driver GymDemo (Main Program)

1. Display the long welcome banner with the gym name
2. Create five Reception objects using carefully chosen test data so that all menu options can be properly tested:
3. Two receptions must have exactly the same pass distribution and the same number of membership cards
4. One reception must have the same total dollar value as another but a different combination of passes, and it must have at least three cards
5. Two receptions must have identical pass distribution (different from the others) and zero membership cards
6. Store these five Reception objects in an array
7. Start the main program loop (keep running until the user chooses option 0)
8. Display the full menu with the decorative border and all nine options plus quit
9. Prompt the user to enter their choice
10. Read the choice and make sure it is between 0 and 9
11. If the user chooses 1
12. Show the complete content of all five receptions (passes + cards)
13. If the user chooses 2
14. Ask which reception number (0–4), validate the input, then display that reception's full details
15. If the user chooses 3
16. Compare every pair of receptions and list those that have the same total dollar amount of passes
17. If the user chooses 4

18. Compare every pair and list those that have exactly the same number of each pass type
19. If the user chooses 5
20. Compare every pair using Reception's equals method and list the equal ones
21. If the user chooses 6
22. Ask which reception, then ask for type, name, day, month → create a new GymCard → add it to the chosen reception
23. If the user chooses 7
24. Ask which reception and which card index → remove that card and inform the user of success or failure
25. If the user chooses 8
26. Ask which reception and card index → ask for new expiry day and month → update the card
27. If the user chooses 9
28. Ask which reception → ask how many of each of the five pass types to add → update and display the new total dollar value
29. If the user chooses 0
30. Exit the loop
31. Otherwise
32. Display a friendly invalid choice message and ask again
33. After the loop ends, display the goodbye message
34. Close the scanner