******

***FatBit***

***Chandler Deloach, Ethan Hendrix, Jayse White***

**----------------------**

**---Use Cases---**

**----------------------**

**Use Case00:**

Scope/System: FatBit device firmware

Name: Count Steps

Level: User-goal

Primary Actor: user/wearer

Stakeholders and Interests:

- User: wants to count steps

Preconditions:

1. Device has power

2. Device is fitted properly

Postconditions: User has knowledge of how many steps they have walked in a given day # Original was not a postcondition, it was a functional requirement related to this use case

Main Success Scenario:

1. User takes steps

2. User requests step count

3. FatBit displays step count

4. User references step count

Extensions:

1.a Device loses power while counting steps:

1 Device will retain step count prior to loss-of-power for the next time it is powered on

2.a User resets step count:

1. FatBit will reset the step count to zero

2. FatBit displays new step count of zero upon users requests

3. FatBit updates step count

Special Requirements: Step count must take into account varying stride lengths

Variation in Technology and Data:

1.a Steps tracked with pedometer

2.a User requests step count with button on device

Frequency of Occurrence: Continuous while powered on and worn

Miscellaneous:

Topics to Research:

- Reliability (i.e. accuracy) of currently deployed pedometer technology

**Use Case01:**

Scope/System: FatBit device firmware

Name: Watch functionality

Level: User goal/System goal

Primary Actor: User

Stakeholders and Interests:

- User: wants to keep track of time

Preconditions: Time has been set by user

Postconditions: User knows the current time

Main Success Scenario:

1. FatBit keeps track of time

2. User references display for current time

Extensions:

1.a Error results in time loss:

1. FatBit requests that user resets current time

Special Requirements:

- Ability to dictate time successfully after a singular reference

- Ability to detect error in timekeeping and automatically reference user for correction

Variation in Technology and Data:

2.a LED display to report time to user

Frequency of Occurrence: Continuously while FatBit has power

Miscellaneous:

1. Possibly add alarm function at some point

**Use Case02:**

Scope/System: FatBit device firmware

Name: Calories Burned Counter

Level: User goal/System goal

Primary Actor: User

Stakeholders and Interests:

- User: wants to record calories burned

Preconditions: Health information must be input (i.e. height, weight, age)

Postconditions: User has gained knowledge of calories burned during the current day

Main Success Scenario:

1. User activates 'exercising' mode

2. Device tracks time spent exercising

3. User requests calorie data from device

4. Device displays calories burned throughout day

Extensions:

1.a FatBit is not in 'exercising' mode:

1. FatBit will capture activity data but may not be as accurate in its reporting

2.a Device loses power during active hours:

1. FatBit will not collect data during this period nor calculate calories burned during this period

3.a FatBit powers on after losing power:

1. FatBit will display last-saved amount of calories burned

Special Requirements:

- Algorithm to estimate calories burned based on health data

- Band made of waterproof and durable plastic

Variation in Technology and Data:

3.1 User requests calorie data by pushing button

4.1 Device displays calories burned in kCal

Frequency of Occurrence: Continuously while powered on and worn

Miscellaneous:

- calculated during active time periods

- Topics to Research:

- determining active heart rate for inactive versus normally active levels (i.e. what is a resting heart rate for user health data)

**Use Case03:**

Scope/System: FatStation Desktop Platform (FSDP)

Name: Activity Statistics View

Level: User goal/System goal

Primary Actor: User

Stakeholders and Interests:

1. User: wants to view activity history and statistics

Preconditions:

- FatStation Desktop Platform must be installed and running on user's PC

- FatBit is connected to user's PC

Postconditions: User is able to make better informed decisions about fitness goals to set based on activity history

Main Success Scenario:

1. Software receives data logs from FatBit device

2. Software verifies integrity of data logs

3. Software generates viewable graphics from data received

4. Software displays graphics to user

5. User interprets statistics

Extensions:

2.a A month or longer has passed since last health update:

1. user is prompted to re-input health information to track changes and keep statistics accurate

3.a Software rejects data logs:

1. Error is raised informing user that the desktop cannot read the device correctly or that the data is corrupted

2. Software suggests steps to prevent recurrences

Special Requirements:

- Graphics are displayed through easy to use GUI

- Graphics need to be well defined and easy to read

- Graphics need to be understandable

- Desktop software must run on x86 Linux

- Data log retention period of thirty days

Variation in Technology and Data:

3.a Data log integrity verification is done through MD5 key

Frequency of Occurrence: Once a month or as desired by user

Miscellaneous:

1. Desktop platform compatibility for other OS to be added in future iterations

**Use Case04:**

Scope/System: FatStation Desktop Platform (FSDP), FatBit device firmware

Name: View progress of meeting fitness goals

Level: User goal/System goal

Primary Actor: User

Stakeholders and Interests:

User: Wants to accomplish personal fitness goals

Preconditions:

- FatStation Desktop Platform must be installed and running on user's PC

- FatBit is connected to user's PC

- User has input basic health information

Postconditions: User is more effectively able to adjust physical activity throughout daily routine to meet fitness goals

Main Success Scenario:

1. User inputs desired fitness goals

2. Software transmits fitness goals to FatBit

3. FatBit tracks and reports progress towards completion of fitness goals

4. After a month or other user-specified period of time, FatBit prompts user to re-connect to pc to update goals

Extensions:

a. FatBit disconnected before goals can be sent to device

1. User prompted to reconnect FatBit before goals can be finalized

2.a FatBit disconnected during transmission of fitness goals to FatBit

1. FatBit warns user of incomplete data transfer

2. FatBit deletes all data from incomplete transfer

Special Requirements:

- Progress towards completion of fitness goals must be able to be determined quickly and easily at a single glance on device

- Graphic that represents percentage towards goal should be easily understandable as such even by new users

Variation in Technology and Data:

- Desktop with x86 linux is required to run FSDP

Frequency of Occurrence:

Once a month or as often as user may desire

Miscellaneous:

1. Monthly average completion rate held on FatBit with long term records kept within FSDP

**--------------------------------------**

**---Operation Contracts---**

**--------------------------------------**

**C00:**

* *Operation:* signalStep()
* *Cross references:* Use Case: Count Steps
* *Preconditions:* 
  + FatBit is properly communicating with accelerometer
* *Postconditions:* 
  + StepCounter has received the call to increment the step count
  + StepCounter has initiated an update to StepLog’s values

**C01:**

* *Operation:* updateStepsToday()
* *Cross references:* Use Case: Count Steps
* *Preconditions:*
  + The accelerometer has signaled a step
* *Postconditions:* 
  + StepCounter has incremented the step count
  + StepCounter has data to write to the StepCountLog

**C02:**

* *Operation:* write( int data )
* *Cross references:* Use Cases: Count steps, Monitor Heart Rate, Calculate Calories Burned
* *Preconditions:*
  + A data source has writable data for a log
* *Postconditions:* 
  + The appropriate Log has written the data provided

**C03:**

* *Operation:* updateMetric( int data )
* *Cross references:* Use Cases: Count Steps, Monitor Heart Rate, Calculate Calories Burned
* *Preconditions:*
  + A Log has a metric for the UILayer to update
* *Postconditions:* 
  + The UILayer has received the metric from the Log
  + The UILayer has updated the view with the new metric

**C04:**

* *Operation:* signalBPM()
* *Cross references:* Use Case: Monitor Heart Rate
* *Preconditions:*
  + The FatBit is properly communicating with the heart rate sensor
* *Postconditions:* 
  + The HeartRateMonitor has received a call to update the heart heart rate
  + HearRateMonitor has initiated an update to the HeartRateLog

**C05:**

* *Operation:* updateHeartRate()
* *Cross references:* Use Case: Monitor Heart Rate
* *Preconditions:*
  + The heart rate sensor has signalled a heart rate
* *Postconditions:* 
  + The HeartRateMonitor updated the current heart rate
  + The HeartRateMonitor has data to write to the HeartRateLog

**C06:**

* *Operation:* calculateCalories()
* *Cross references:* Use Case: Calculate Calories
* *Preconditions:*
  + CalorieCalculator has received a BPM from the HeartRateLog
* *Postconditions:* 
  + The CalorieCalculator has calculated the calories burned
  + CalorieCalculator has data to write to the CalorieLog

**C07:**

* *Operation:* setGoals()
* *Cross references:* Use Case: Set Goals
* *Preconditions:* 
  + none - this is the first operation
* *Postconditions:* 
  + FatStation has initiated the process to set goals
  + Log objects have been created
  + A Goals object has been created

**C08:**

* *Operation:* enterGoals()
* *Cross references:* Use Case: Set Goals
* *Preconditions:*
  + Setting goals is underway
* *Postconditions:* 
  + The user has entered a goal
  + A corresponding Goal object has been created

**C09:**

* *Operation:* setGoal()
* *Cross references:* Use Case: Set Goals
* *Preconditions:*
  + Setting goals is underway
  + The user has entered a goal
* *Postconditions:* 
  + The goal object has been associated with the Goals object

**C10:**

* *Operation:* seeProgress()
* *Cross references:* Use Case: Set Goals
* *Preconditions:*
  + The User has set goals
* *Postconditions:* 
  + The Goals object has initiated a read from the Logs

**C11:**

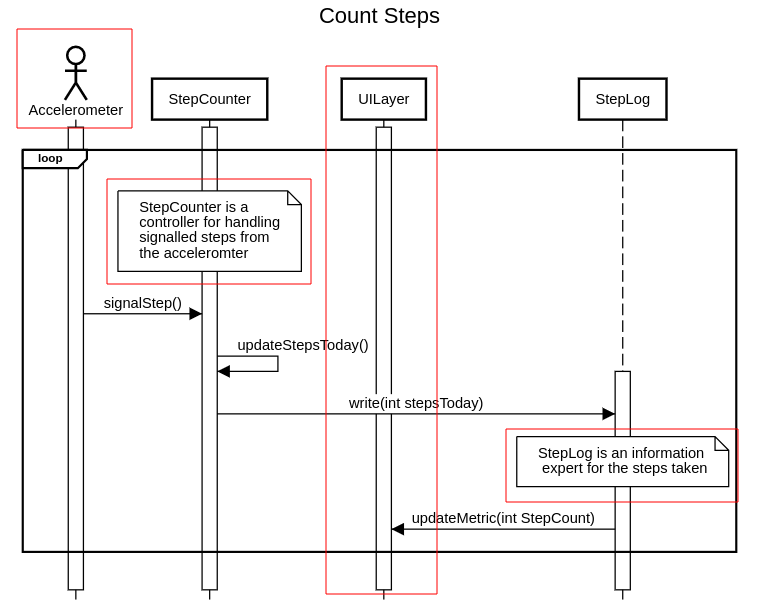
* *Operation:* calculateProgress()
* *Cross references:* Use Case: Set Goals
* *Preconditions:*
  + The User has requested their progress
* *Postconditions:* 
  + The Goals object has read from the Logs
  + The Goals object has updated it progress attributes for each goal

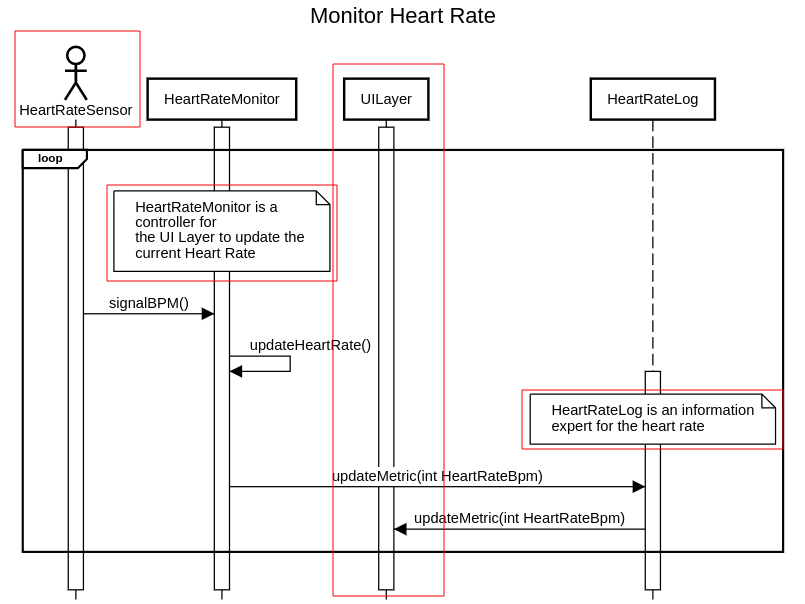
**C12:**

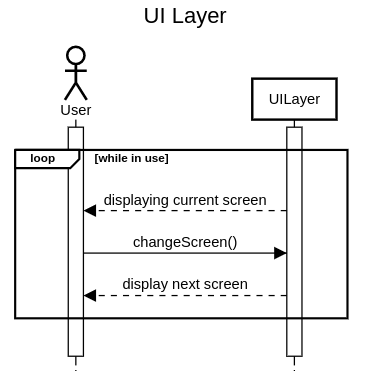
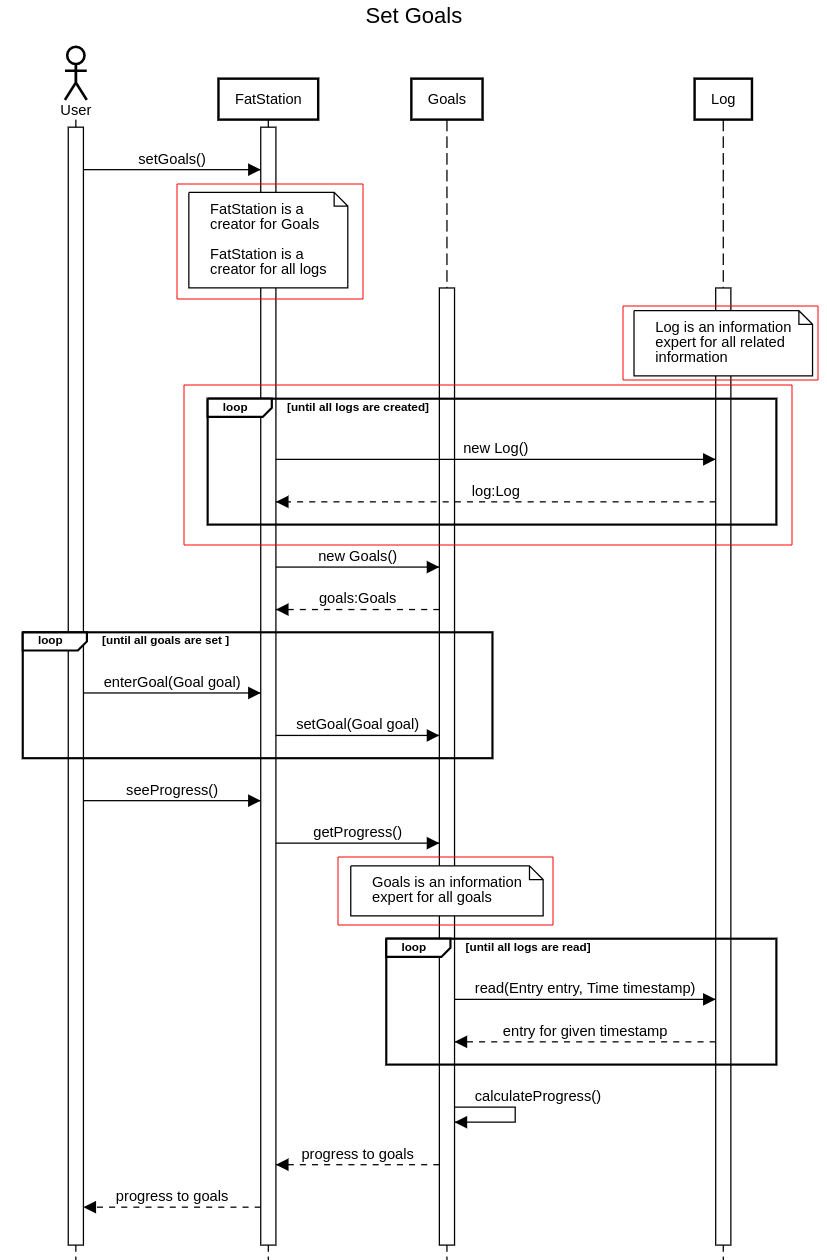
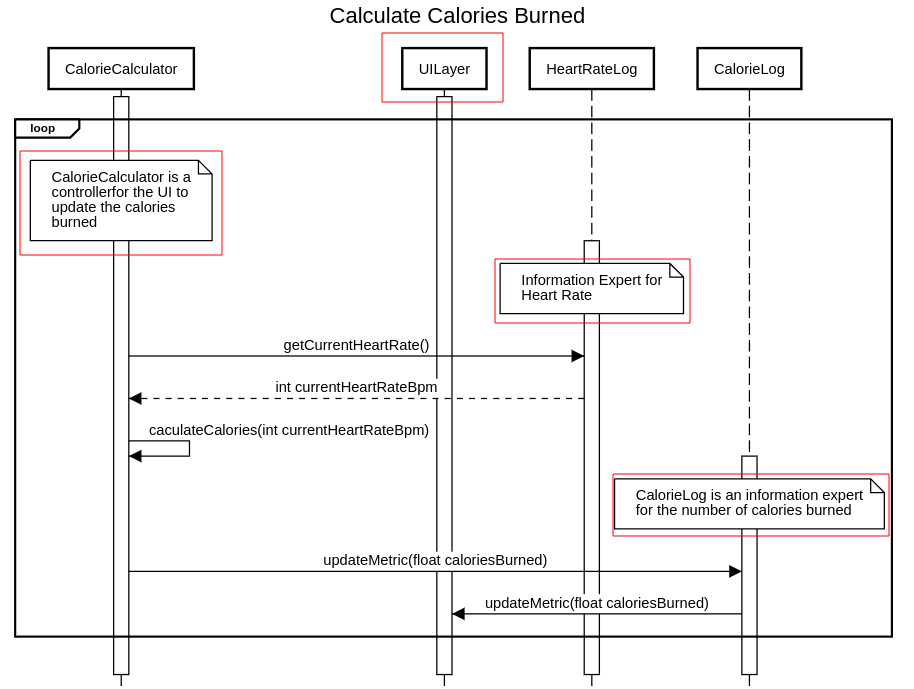
* *Operation:* read( Log log )
* *Cross references:* Use Cases: Calculate Calories, Set Goals
* *Preconditions:*
  + There is data in the log to read
* *Postconditions:* 
  + The calling object has received data from the appropriate log

**-----------------------------------**

**---Sequence Diagrams---**

**-----------------------------------**

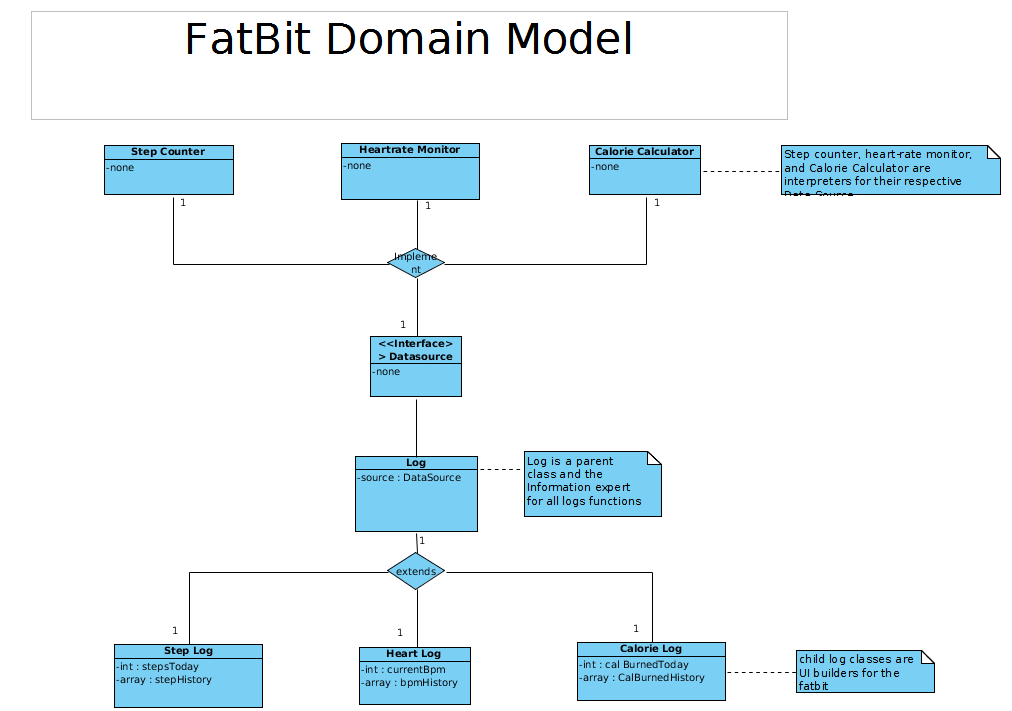
****

****

**-----------------------------**

**---Domain Model---**

**-----------------------------**

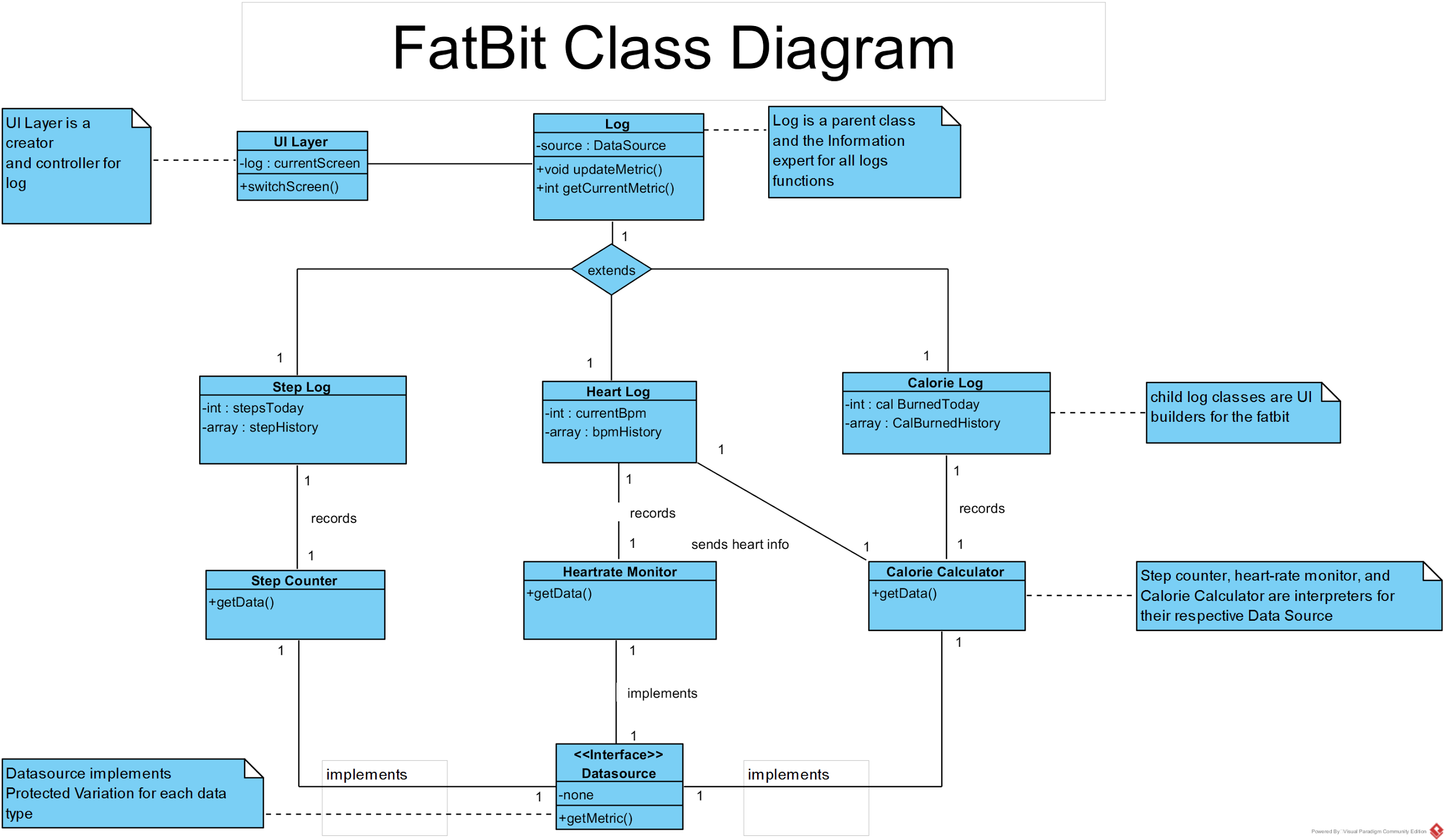
****

**----------------------------**

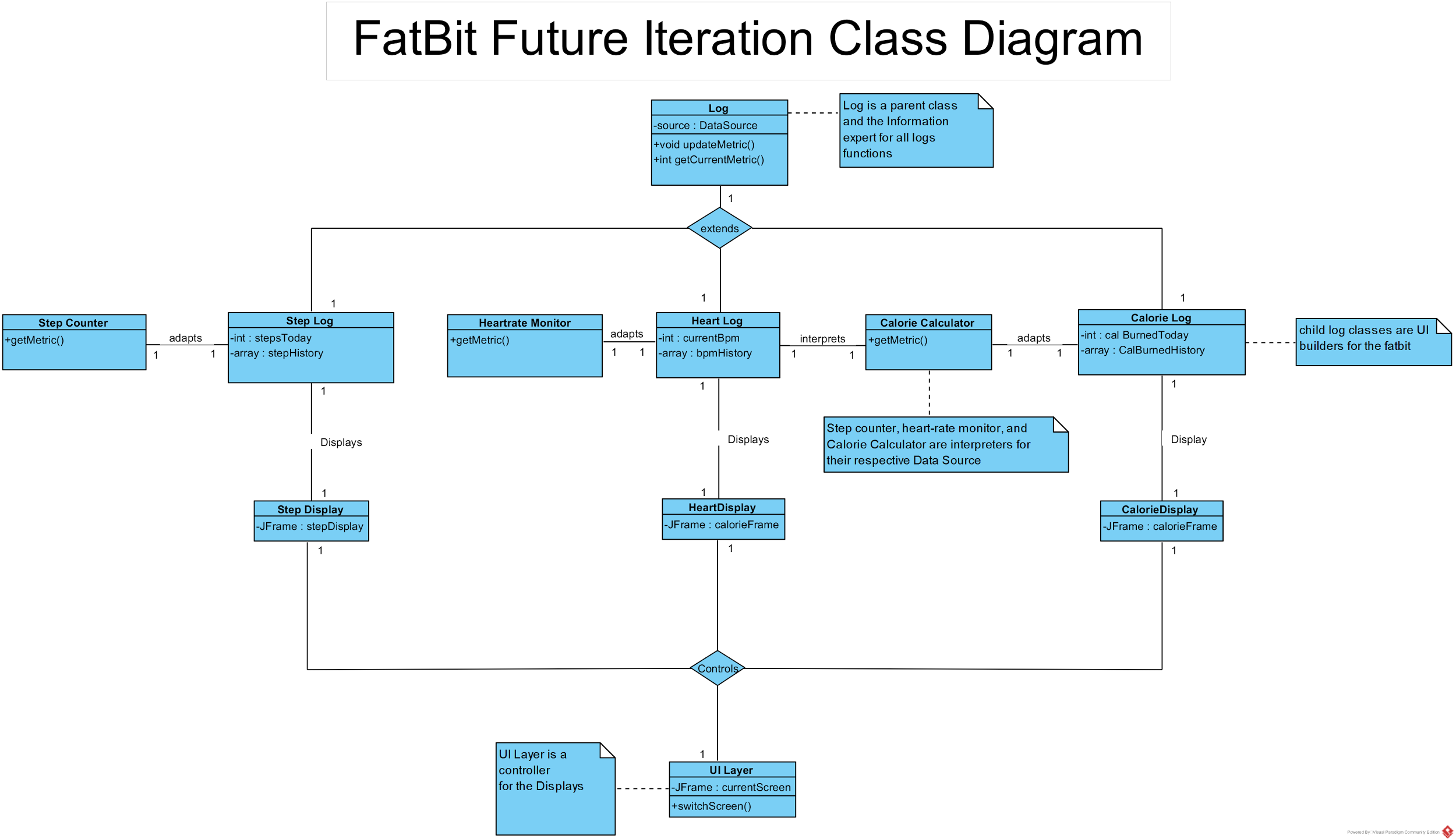
**---Class Diagram---**

**----------------------------**

*Current Implementation*

****

*Possible Implementation for the Future*

****

**--------------------------------------------------**

**---Design Patterns and Decisions---**

**--------------------------------------------------**

* *Log*
  + Information Expert
    - Each log is responsible for knowing its values (i.e. HeartRate log knows the heart rate)
  + Builder
    - Each log is responsible for building its corresponding UI element
* *UILayer*
  + Controller
    - The UILayer handles system events and responds accordingly to user input
* *Datasources*
  + Adapter
    - The interface simplifies the communication and allows the pieces to talk to each other

**--------------------------**

**---Requirements---**

**--------------------------**

* Functional
  + Count steps
  + Track user heart rate
  + Calculate calories burned
  + Display current time
* Non-Functional
  + Efficient utilization of resources
  + Operate on a device with limited memory space
  + Information viewable on small screen
  + Clean display aesthetic

**-------------------**

**---Analysis---**

**-------------------**

* Calculate calories
  + Based on heart rate
* Implemented GUI in Java Swing
  + Minimal buttons for clean interface
  + Tested in small window
* Generic approach with Inheritance
  + Data source separation through inheritance
* Log as a parent class
  + Each log in our FatBit is a child of the parent log class