Technical Doc: Module 5

# ***Classes, Objects, Constructor, Instance variables and methods, Overloading***

Classes:

**Blueprints for objects. Objects are constructed out of classes.**

* Holds methods and variables exclusive to those objects (encapsulation)
* Can have parameters for how an object is constructed (construction methods)
* Can have instances of objects
  + Each object can have different variables (instance variables)
* Sometimes classes can have their own unique folder if they are public static

A screen shot of a computer program

AI-generated content may be incorrect.

*The athlete class. It has a constructor, “public Athlete” and class specific methods that work with athlete objects like getAge which returns the value attatched to a private variable within the class and inCal which creates an array to be attached to each athlete.*

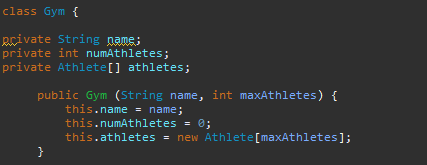
Objects + Constructors:

**Objects are constructed out of classes (blueprints). Think of objects as iterations of a class. (Instantiation)**

* Each object can hold its own data field (variables).
  + The variables are called instance variables
* Can be accessed by methods within their own class

**Constructor**

* A method that instantiates a new instance of an object
* Parameters within a constructor method can attach values to the data field within an object
  + Instances of variables within an object
* The constructor initializes the object, which creates space on the heap
* Constructors must always be named the same as the class
* The keyword “new” accesses the constructor effectively constructing a new object



*Public Gym is the constructor method, notice it has the same name and naming conventions as the Gym class itself. It has the parameters of (name, maxAthletes) and gives the object being constructed a new instance of 3 variables: name, numAthletes, and athletes. (Denoted by “this.” Keyword)*



*Object being constructed. (Denoted by “new” keyword) This Gym object is being constructed with the name of “Elite Fitness” and a max of athleteQuant, which refers to an array object, which means the maxAthletes value is the size of an array of athletes.*

Instance Variables + Methods:

*Instance Variables*

* Denoted by “this.” Keyword
* Every object has these variables, but they likely have different values within each field
* Could be non-primitive or primitive
* Only exist in memory with objects
  + Objects must have been initialized for instances of variables to be allocated in memory
* Most defined within constructor
* Operates within heap (attached to objects)

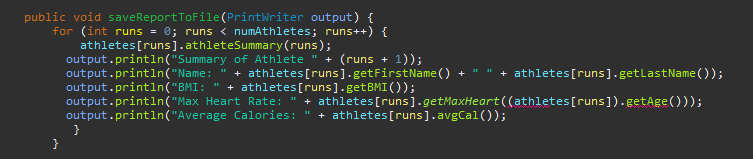
A black background with colorful text

AI-generated content may be incorrect.

*Instance variables within a constructor. Denoted by “this.”, these variables have an initial value of null, and they will only have a value allocated for in memory when an object is created with a new instance.*

*Instance Methods*

* Exact same characteristics as instance variables
* Can access instance variables
* Operates with objects’ instances
* Operates within heap (attached to object



*Instance method. It will print out a summary of calculations utilizing instance variables, or instance variables themselves, all of which are attached to an object. This method works on objects themselves, not the class, and uses “.” to be called upon.*

Overloading:

**Overloading refers to overloading a method. Multiple methods will exist under the same name but have different method signatures.**

* Different return types or different parameters



*There are two “positiveInput” both public and static. They both receive the validinput value within their method signature. Both methods check to make sure an input value is positive. However, they have different return types and data typers, one possesses a data type of double within its signature, the other possesses an integer data type in their method signature. One verifies an integer, the other verifies a double.*

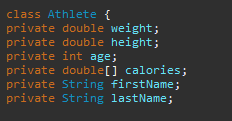
# ***Object instantiation + accessing object's data fields and methods***

Object Instantiation:

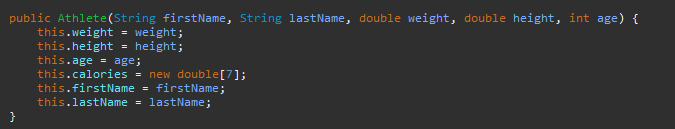
* A new instance of an object is created
  + Like a new instance of a variable
* This is denoted by the “new” keyword
* This new instance of an object within a class may have new instances of variables

Access object data fields and methods:

* Object data fields can sometimes be private within its class (encapsulated)
* To access private data fields getters and setters must be used
* Private data fields can be turned into instance variables if declared within their respective class – This makes it so that the initial data fields cannot be edited, but new instances can be edited
  + Constructor method can edit these values

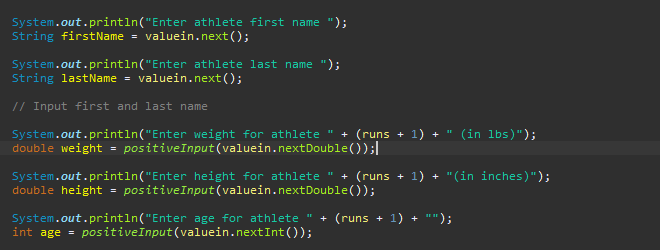


*Initial private encapsulated variables are created unique to the athlete class (private).*



*Constructor method creates new instances of the private variables within the new athlete object (this.)*



*A new athlete object is constructed at index [runs], and with its creation, new instances of the variables are also created. This is due to the Athlete constructor method having those variables within its perimeter. *

*The values of the new instance variables are inputted by the user.*

# ***Encapsulation, getters and setters***

Encapsulation:

**Encapsulated data fields (variables), encapsulated methods, etc. are all enclosed within a class. These traits can only be accessed within the class they are enclosed in, unless in some cases, getters or setters are used.**

*Private variables and methods*

* The “private” keyword encapsulates variables and methods
  + Encapsulated variables and methods
    - Can only be accessed within the class
    - Getters and setters can be used to access a modified version of the private variable
    - Encapsulated methods cannot be accessed outside of class

A screen shot of a computer code

AI-generated content may be incorrect.

*These private variables are variables encapsulated within the Athlete class. They can only be accessed either in the class, or outside of the class via getters and setters like setAge or getAge.*

Getters and Setters:

*Getters*

* Method to return a value
  + Returns a value within a private variable
  + Allows access to private variables outside of class
  + Getters must be defined within the class of private variables



*The “getMaxHeart” method requires age, so to access the private variable age of the object at Athletes[runs] the “getAge” method must be used to return the value attached to age. (The “.” In “.getAge()” refers to accessing a method and using it on an object.*

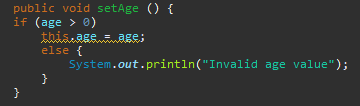
A screen shot of a computer

AI-generated content may be incorrect.

*The “getAge” method returns the age, and since age is an integer, the return type is int.*

*Setters*

* Method to edit a value
  + Edits a value within a private variable
  + Allows access to private variables outside of class
  + Setters must be defined within the class of private variables

**

*The set age method checks to make sure if the age value is greater than 0. If it is, the private variable age’s new value will be set. If not, it will not be set, the console will print “Invalid age value” instead.*



*This utilizes the set age method in order to change the private variable age within main.*

# ***Object reference variables + memory on stack and heap***

Reference variables:

* Reference variables are essentially addresses
  + Addresses that direct to a certain spot on the heap
    - Reference variables are stored on stack
    - “Spot on heap” refers to a certain object on the heap



“Gym gym” – Reference variable

“new Gym(“Elite Fitness”, athleteQuant)” – Object being constructed with new instances

“Gym gym” directs to the spot on the heap where “new Gym” resides

Stack + Heap:

* Heap
  + All objects are stored on the heap
    - Arrays, new gyms, new athletes
      * Initial space is allocated for array size
  + All instance variables are stored on the heap
    - New instances are created with new objects
* Stack
  + Methods, reference variables, primitive variables are stored on the stack
    - Once the code has run, the stack will then collapse
      * Space isn’t allocated until methods or local variables are initialized
        + Uninitialized variables are null valued

# ***Array of objects***

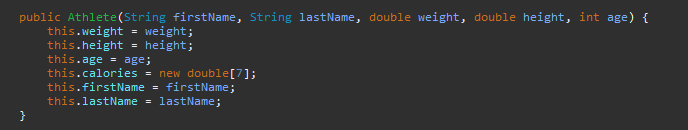
Object Arrays:

* Arrays can hold objects
  + Promotes DRY – reduces repeatedly initializing objects
  + Each object can hold instances of variables
  + Can be used within a loop
  + Drastically improves code’s readability



*Updates the athlete object at x index with a new instance*

* *New instances of variables*
* *Saves data to each object – Each index has its own instance of variables*



*Constructor method*

* *The method that will create a new instance of an object*
  + *“this.” Refers to “this instance”*



*The creation of the array with athlete objects*

* *Athlete[] athletes – reference variable*
* *Athlete - class*
* *new Athlete [athleteQuant] – object being constructed (hint: “new” keyword)*

A computer screen shot of a program code

AI-generated content may be incorrect.

*An array of objects being used within a loop*

* *the array of calories has 7 indexes – the for loop will run 7 times*
* *For each day, the variable “calBurn” will be updated with a new input*
* *The calories array within each athlete at index x will effectively updated*