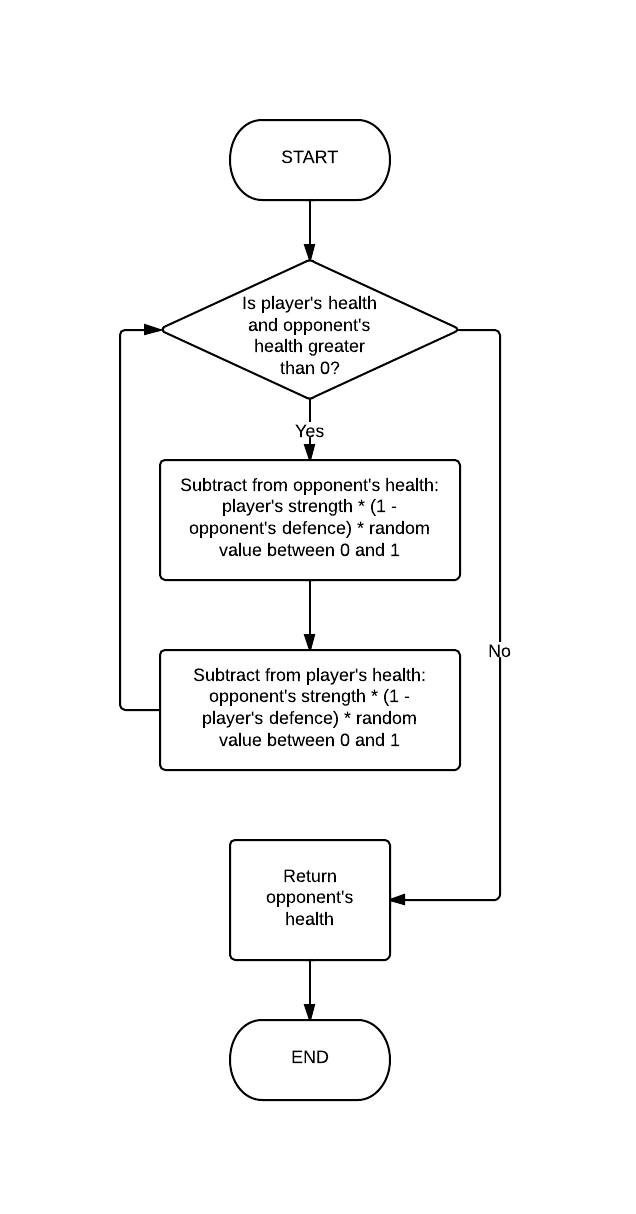
2.2.2 Monster

Monster - class containing information about a single monster.

Monster class contains following attributes:   
-id:int - ID of the monster (randomly generated String)  
-name:String - name of the monster  
-dob:Date - monster's date of birth  
-dod:Date - monster's date of death  
-baseStength:double - strength of the monster, used during breeding  
-currentStrength:double - strength of the monster, used during fighting  
-baseHealth:double - health of the monster, used during breeding  
-currentHealth:double - health of the monster, used during fighting  
-fertility:float - fertility of the monster  
-userID:String - ID of the owner  
-saleOffer:int - if other than 0, the monster is offered for sale  
-breedOffer:int - if other than 0, the monster is offered for breed  
-serverID:int - ID of the server on which the monster exists  
-MAX\_CHILDREN:int - maximum number of monsters that can be result of breeding

Monster class contains following methods:   
+fight(opponent:Monster):double - contains fighting algorithm. Takes the opponent monster as a parameter. Returns opponent's health.   
+breeding(other:Monster):Monster[] - contains breeding algorithm. Takes the monster to breed with as a parameter. Returns array of new monsters that are the result of breeding.   
+updateStats(strength:double, defence:double, health:double):void - updates statistics of the monster. 5.2.1 Battling



public double fight**(**Monster opponent**)**

**{**

Random randomGenerator **=** **new** Random**();**

double random **=** randomGenerator**.**nextDouble**();**

**while(this.**currentHealth **>** 0 **&&** opponent**.**currentHealth **>** 0**)** **{**

opponent**.**currentHealth **-=** **this.**currentStrength **\*** **(**1**-**

opponent**.**currentDefence**)** **\*** random**;**

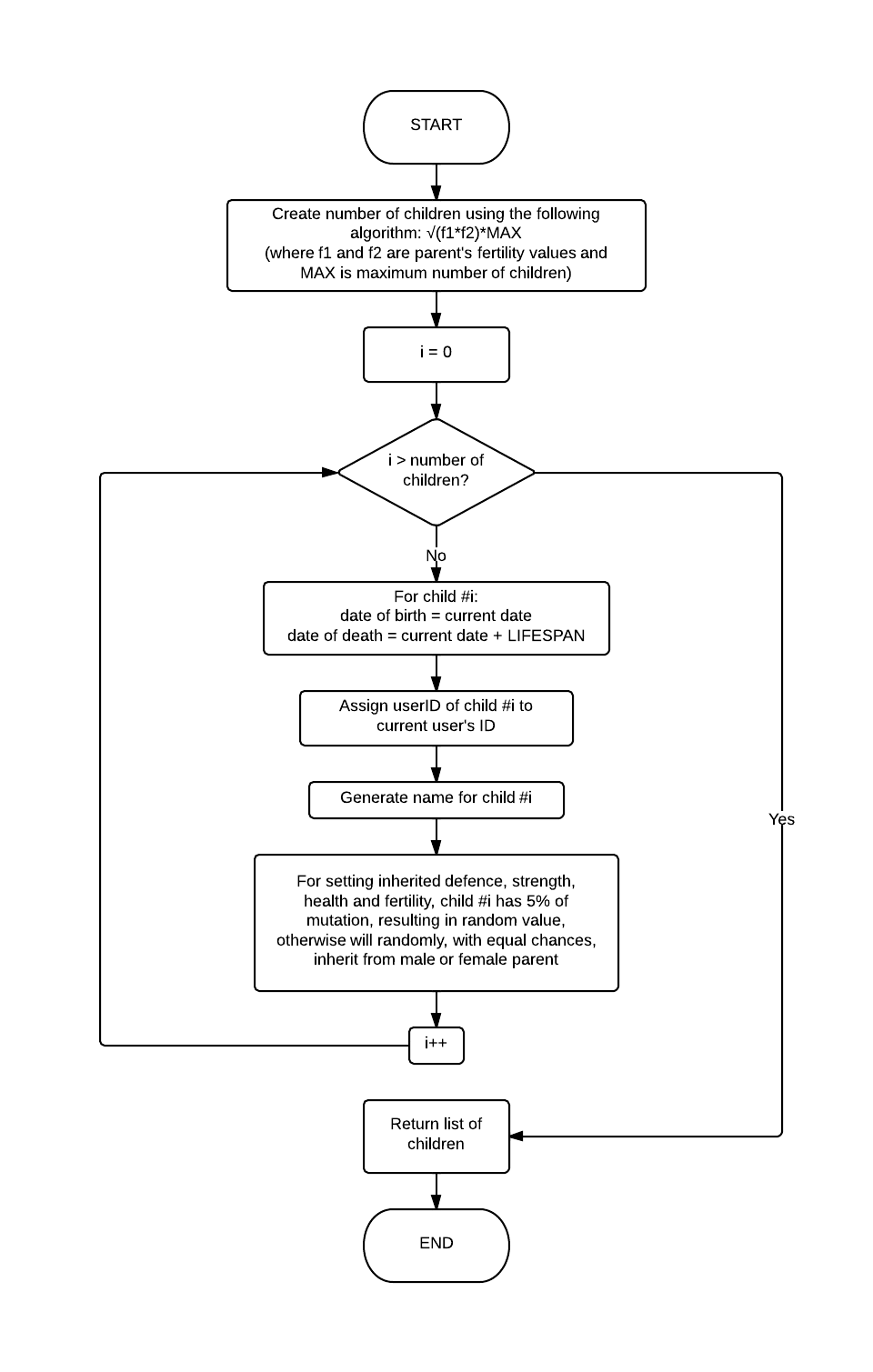
**this.**currentHealth **-=** opponent**.**currentStrength **\*** **(**1**-**

**this.**currentDefence**)** **\*** random**;**

**}**

**return** opponent**.**currentHealth**;**

**}**5.2.2 Breeding



public Monster**[]** breeding**(**Monster other**)** **{**

Random r **=** **new** Random**();**

int numberofchildren **=** **(**int**)** **(**Math**.**sqrt**(**fertility **\*** other**.**fertility**)** **\*** MAX\_CHILDREN**);**

Monster**[]** children **=** **new** Monster**[**numberofchildren **+** 1**];**

**for** **(**int i **=** 0**;** i**<=** numberofchildren**;** i**++){**

children**[**i**]=new** Monster**();**

children**[**i**].**id **=** "0"**;**

children**[**i**].**dob**=new** Date**();**

children**[**i**].**dod **=** **new** Date**(**children**[**i**].**dob**.**getTime**()+**LIFESPAN**);**

//this is assuming that the children go to the owner of the monster

that calls the method

children**[**i**].**userID **=** **this.**userID**;**

children**[**i**].**name **=** NameGenerator**.**getName**();**

//generating inherited defense

**if(**r**.**nextInt**(**100**)<**5**){**

children**[**i**].**baseDefence**=**r**.**nextDouble**();**

**}** **else** **if(**r**.**nextInt**(**100**)<**50**){**

children**[**i**].**baseDefence**=**baseDefence**;**

**}** **else** **{**

children**[**i**].**baseDefence**=**other**.**baseDefence**;**

**}**

children**[**i**].**currentDefence **=** children**[**i**].**baseDefence**;**

//generating inherited strength

**if(**r**.**nextInt**(**100**)<**5**){**

children**[**i**].**baseStrength**=**r**.**nextDouble**();**

**}**

**else** **if(**r**.**nextInt**(**100**)<**50**){**

children**[**i**].**baseStrength**=**baseStrength**;**

**}** **else** **{**

children**[**i**].**baseStrength**=**other**.**baseStrength**;**

**}**

children**[**i**].**currentStrength **=** children**[**i**].**baseStrength**;**

//generating inherited health

**if(**r**.**nextInt**(**100**)<**5**){**

children**[**i**].**baseHealth**=**r**.**nextDouble**();**

**}else** **if(**r**.**nextInt**(**100**)<**50**){**

children**[**i**].**baseHealth**=**baseHealth**;**

**}** **else** **{**

children**[**i**].**baseHealth**=**other**.**baseHealth**;**

**}**

children**[**i**].**currentHealth **=** children**[**i**].**baseHealth**;**

//generating inherited fertility

**if(**r**.**nextInt**(**100**)<**5**){**

children**[**i**].**fertility**=**r**.**nextFloat**();**

**}** **else** **if(**r**.**nextInt**(**100**)<**50**){**

children**[**i**].**fertility**=**fertility**;**

**}** **else** **{**

children**[**i**].**fertility**=**other**.**fertility**;**

**}**

**}**

**return** children**;**

**}**