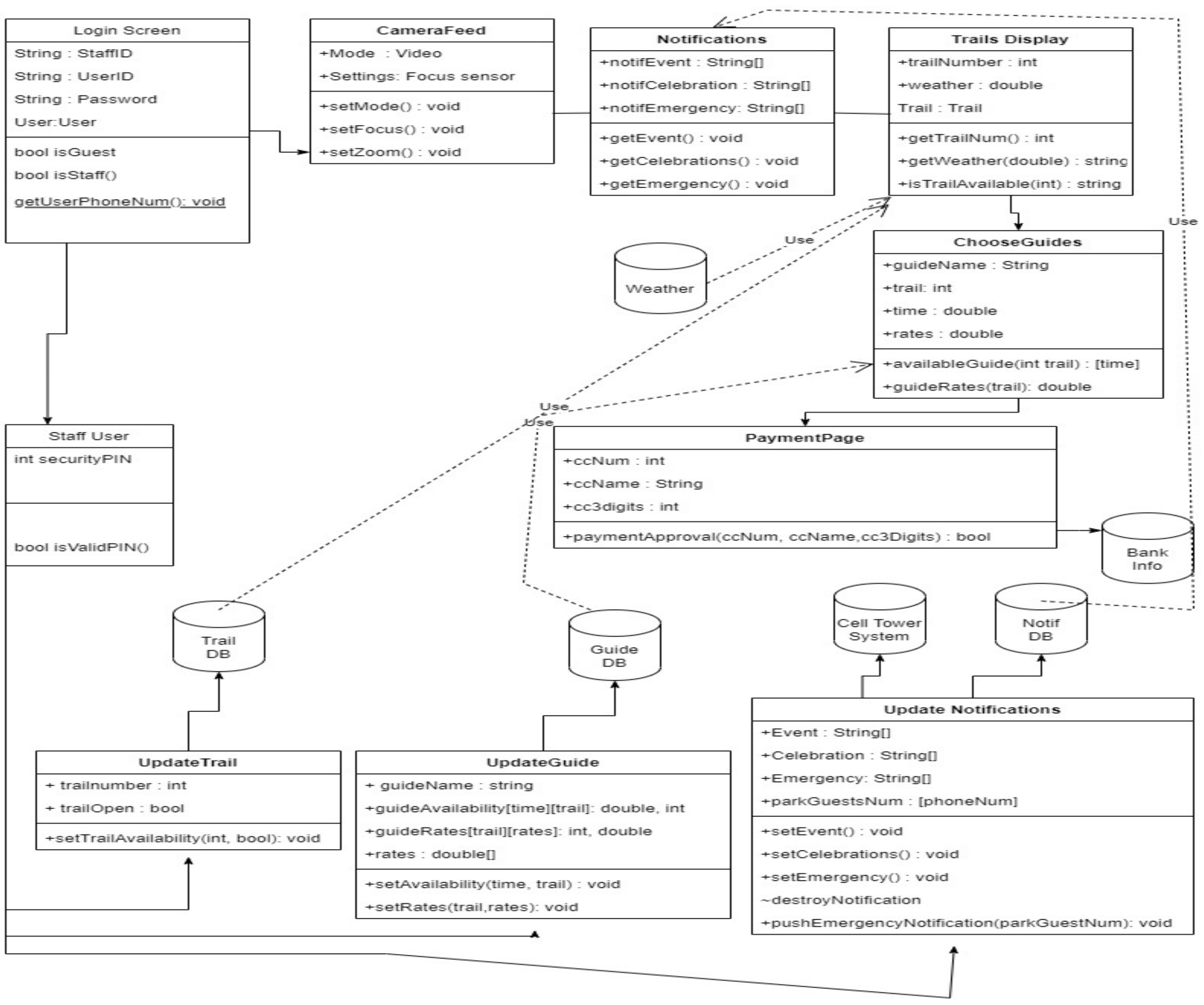
SDS: Test Plan of Kilimanjaro Trekker System

By Chloe, Burak, David

UML Class Diagram



We have made some changes to our original UML class diagrams as we decided to change some functions parameters and return types to help the testing process. Changes were made to the following classes: TrailDisplay, ChooseGuides, and UpdateGuide. In addition some arrows were changed to show how certain classes only use database information.

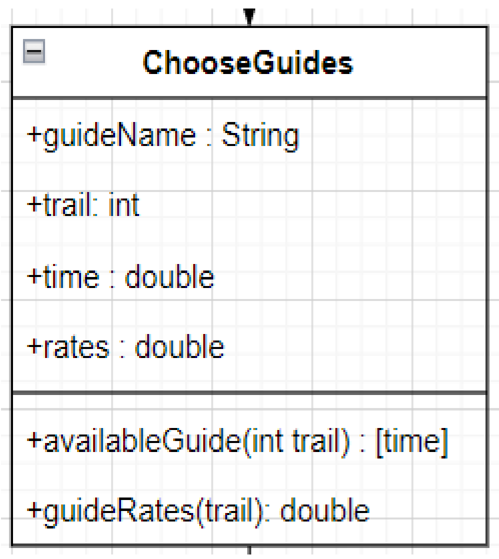
First for trailDisplay, the getTrailNum function was changed from void to int as the function should “get” and return a trail number. The getWeather functions return type was changed from void to a string. Also it was given a parameter of a double weather value. This function should take in a weather value in degrees Fahrenheit, and return a string park guests can understand such as “Cold”, “Hot”, or “Good Weather”. Finally the selectTrail(int) function was removed and replaced with the isTrailAvailable function. This new function takes in an integer as input and returns a string letting the user know if the corresponding trail number is open or not.

Second, for the ChooseGuides class, the guideTimes[][] attribute was removed as the availableGuide function performed a similar role. The AvailableGuide function was also changed in that it now takes in only a trail number input and returns a 1-dimensional array of time values. Before it returned a 2-dimensional array with trails and time, but we believe a park gust should only be able to see the times for the one trail they're interested in.

Lastly, in the updateGuide class there was a removeAvailability function that was removed as the setAvailability function can perform the same task.

Unit Tests

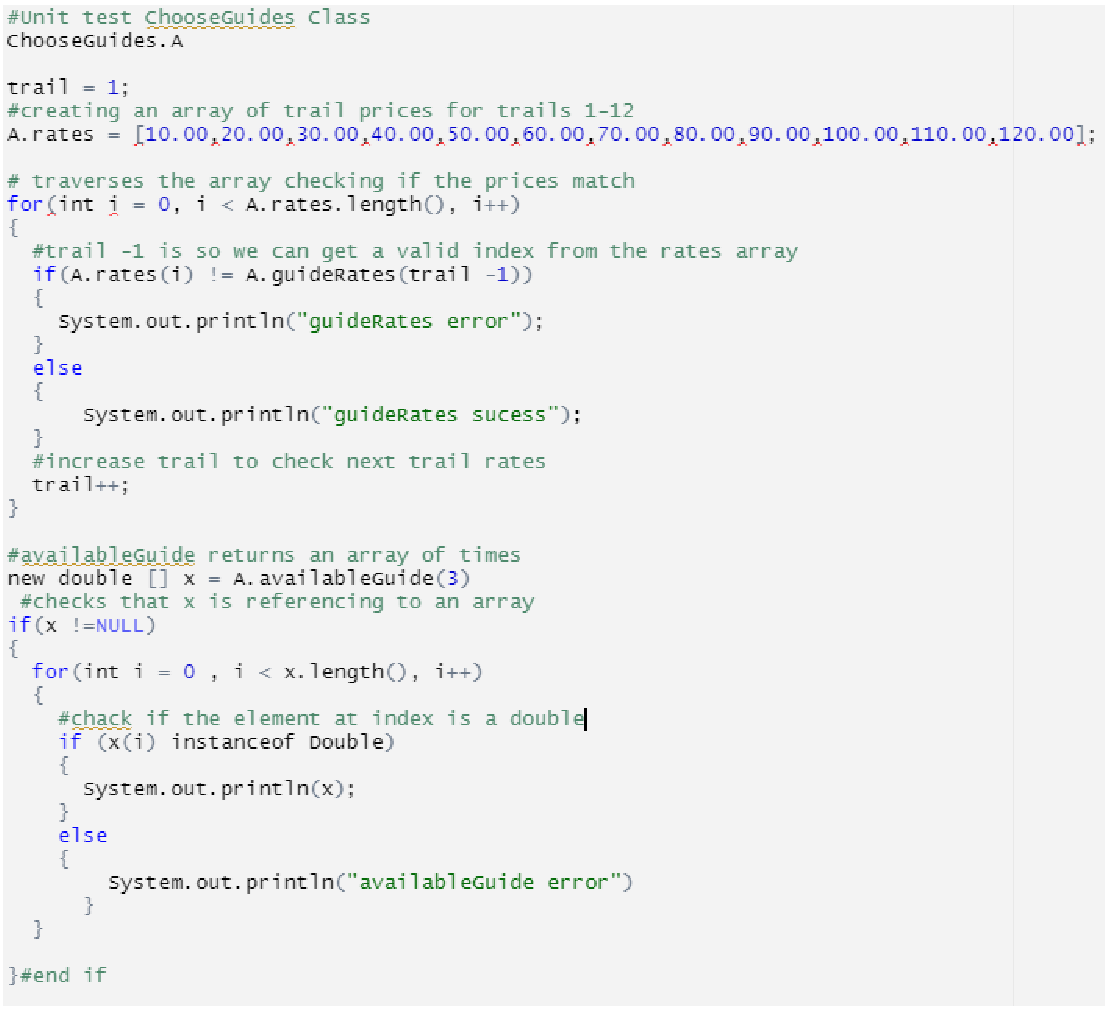
1. **ChooseGuides class**



To test this class must first create a ChooseGude object. Then we can then create an array of rates for the class to use for testing the guide rates function.

The guideRates function takes in an integer representing a trail number and returns the price or Rate for a guide for the specified trail. We can test this function by comparing the function's returned price with a variable that holds a similar price. If the prices match then we know the function returns a price successfully.

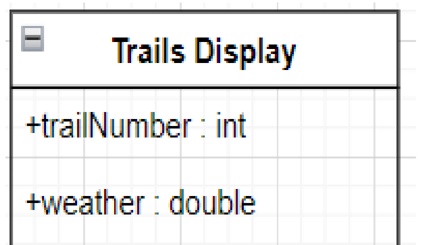
The second function to test is the AvailableGuide() function which again takes in an integer that represents a trail number and returns an array of available times for that trail. To test this function we first call the method and store its array into a variable. Then we first check that the stored array isn't empty. If it has values, we use "instanceof" to check that all numbers are double.



```
#Unit test ChooseGuides Class
ChooseGuides.A

trail = 1;
#creating an array of trail prices for trails 1-12
A.rates = [10.00,20.00,30.00,40.00,50.00,60.00,70.00,80.00,90.00,100.00,110.00,120.00];

# traverses the array checking if the prices match
for(int i = 0, i < A.rates.length(), i++)
{
  #trail -1 is so we can get a valid index from the rates array
  if(A.rates(i) != A.guideRates(trail -1))
  {
    System.out.println("guideRates error");
  }
  else
  {
      System.out.println("guideRates sucess");
  }
  #increase trail to check next trail rates
  trail++;
}

#availableGuide returns an array of times
new double [] x = A.availableGuide(3)
 #checks that x is referencing to an array
if(x !=NULL)
{
  for(int i = 0 , i < x.length(), i++)
  {
    #chack if the element at index is a double
    if (x(i) instanceof Double)
    {
      System.out.println(x);
    }
    else
    {
        System.out.println("availableGuide error")
      }
  }

}#end if
```

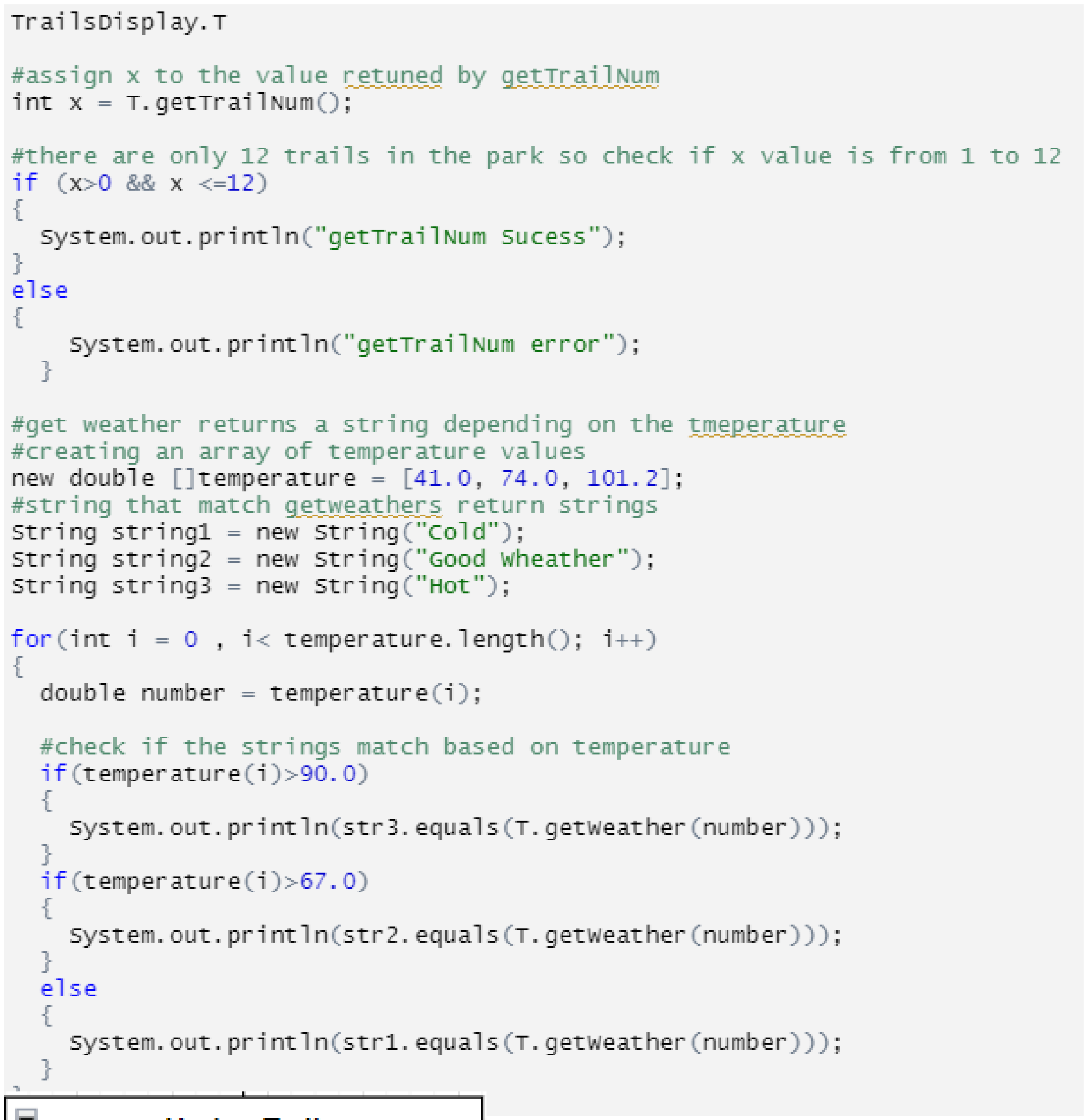
2. TrailsDisplay class

To test this class, we first must make a TrailsDisplay object.

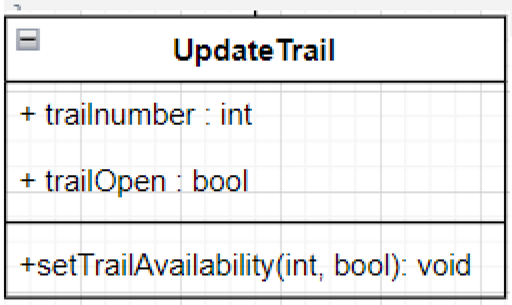


First we call the getTrailNum() function and store its returned integer into a variable. We then check if this value is equal to any number between 1 and 12 because the park only has 12 trails. The function succeeds if the number is between 1 and 12.

Next we test the getWeather function which takes in a double value of weather in fahrenheit and returns a string that the user can understand. For example if the temperature is below 67 degrees F the function returns “Cold”. If the temperature is above 90 degrees the function returns “Hot”. If the weather is between 67 and 90, the function returns “Good Weather”. To test we first create strings identical to the functions output and create an array of temperature values. A for-loop is used to traverse the temperature array and then an if statement is used to check the weather range. Next, the strings we created are compared with the functions returned string. If they are the same, the function passed.



```
TrailsDisplay.T

#assign x to the value retuned by getTrailNum
int x = T.getTrailNum();

#there are only 12 trails in the park so check if x value is from 1 to 12
if (x>0 && x <=12)
{
  System.out.println("getTrailNum Sucess");
}
else
{
    System.out.println("getTrailNum error");
  }

#get weather returns a string depending on the tmeperature
#creating an array of temperature values
new double []temperature = [41.0, 74.0, 101.2];
#string that match getweathers return strings
String string1 = new String("Cold");
String string2 = new String("Good Wheather");
String string3 = new String("Hot");

for(int i = 0 , i< temperature.length(); i++)
{
  double number = temperature(i);

  #check if the strings match based on temperature
  if(temperature(i)>90.0)
  {
    System.out.println(str3.equals(T.getWeather(number)));
  }
  if(temperature(i)>67.0)
  {
    System.out.println(str2.equals(T.getWeather(number)));
  }
  else
  {
    System.out.println(str1.equals(T.getWeather(number)));
  }
```



UpdateTrail

+ trailnumber : int
+ trailOpen : bool

+setTrailAvailability(int, bool): void

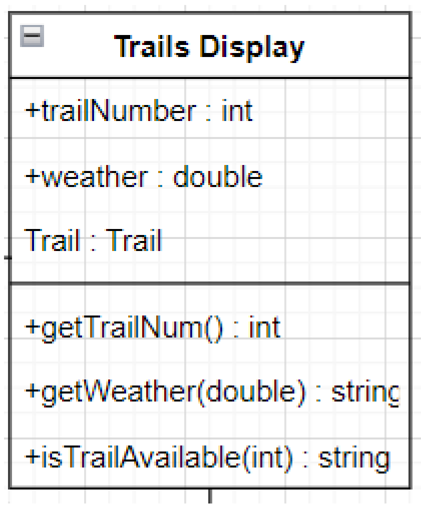
trails > trailsDisplay

Integration Tests

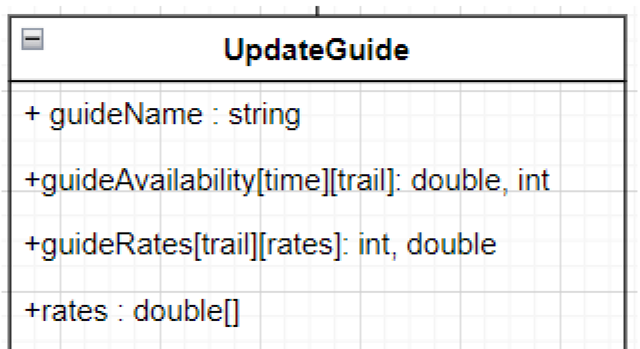
1. **Update**

The updateTrail class is a class that is accessed only by park staff and is used to update the availability of a park trail. The updates should be visible to the park guests in the TrailsDisplay class which will inform them if the inputted trail is open.

To test the integration of the classes, we first create an UpdateTrails object called Ranger and use the setTrailAvailability function to update all 12 trails. We then create strings identical to the isTrailAvailable functions output. If these strings match, then the updates and integration worked as expected.



```
UpdateTrials.Ranger;
TrailsDisplay.Trail;

#updating all 12 trails
Ranger.setTrailAvailability(1,True)
Ranger.setTrailAvailability(2,True)
Ranger.setTrailAvailability(3,True)
Ranger.setTrailAvailability(4,True)
Ranger.setTrailAvailability(5,True)
Ranger.setTrailAvailability(6,True)
Ranger.setTrailAvailability(7,False)
Ranger.setTrailAvailability(8,False)
Ranger.setTrailAvailability(9,False)
Ranger.setTrailAvailability(10,False)
Ranger.setTrailAvailability(11,False)
Ranger.setTrailAvailability(12,False)



String string1 = new String("Available");
String string2 = new String("Not Available");

for(int i = 0 , i < 12; i++)
{
  #using i +1 so that we send a valid trail number from 1 to 12
  #assigning x to the string that isTrailAvailable returns
  String x = Trail.isTrailAvailable(i+1);

  #check if the strings match
  if(x.equals(string1) || x.equals(string2))
  {
    System.out.println("Trail updated suscces");
  }
  else
  {
    System.out.println("Trail update error";
  }
}
```


UpdateGuide
+ guideName : string
+guideAvailability[time][trail]: double, int
+guideRates[trail][rates]: int, double
+rates : double[]

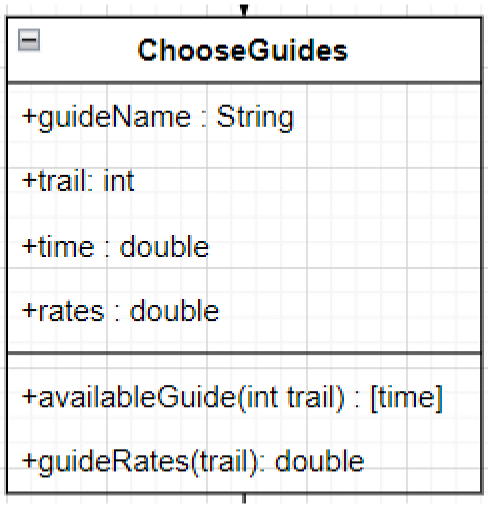
2. updateGuides > ChooseGuides

The updateGuidesl class is a class that is accessed only by park staff and is used to update to availability and rates for a guide of a park trail. The updates should be visible to the park guests in the ChooseGuides class which will inform them of guide price and availability.

First we create an UpdateGuide and ChooseGuides objects and then call the updateGuides class setAvailability function to set guide times for a trail. Then the setRates function is called to set the price for 3 different trails.

Much like in the unit test for the chooseGuides function, we first check if availableGuide returns a non-empty array and check that the values of the array are time double values. If the values are valid, the update was successful.

Lastly the guide rates method returns the price for a trails guide. We compare this value to our expected updated price. If the values match, the price was updated successfully.



```
UpdateGuide.Guide;
ChooseGuides.G;

Guide.setAvailabitiy(13.00, 1);
Guide.setAvailabitiy(14.00, 1);
Guide.setAvailabitiy(15.00, 1);

Guide.setRates(1,10.00);
Guide.setRates(2,20.00);
Guide.setRates(3,30.00);

#available guide returns an array of times available for a trail
double []x = G.availableGuide(1)

 #checks that x is referencing to an array
if(x !=NULL)
{
  for(int i = 0 , i < x.length(), i++)
  {
    #chack if the element at index is a double
    if (x(i) instanceof Double)
    {
      System.out.println(x);
    }
    else
    {
        System.out.println("availableGuide update error")
      }
  }


double a = G.guideRates(1);
double b = G.guideRates(2);
double c = G.guideRates(3);

if(a != 10.00)
{
  System.out.println("guideRates update error")
}
if(b != 20.00)
{
  System.out.println("guideRates update error")
}
if(c != 30.00)
{
  System.out.println("guideRates update error")
}
```

System Tests

The system will show what happens when an emergency occurs in the park and everything will be closed down. In order to update the tour information to apply changes to the data, a user will be created which will be an administrator so that can call **setTrailAvailablity().** According to the emergency situation relevant functions can be called.

newUser(String, int, bool)

-This function provides to create a new user,

-It will also have 3 arguments that represent; string for the username, int for the

-password, and lastly a bool will specify the type of user- either admin or not.

-For instance, (Tim Pence, 789456123, true) New admin user

updateTourl(tourCompany, trialName, tourTime, tourDay, tourMonth, tourYear)

This function will only be accessed by admin users,

It will receive 6 arguments, such as company, trialName, tourDay, tourMonth, tourYear,

When tourCompany is received the companyClass will be called for verification of the company, then company information will be returned.

For instance, if(tourCompany is valid) enter trial name, time, day, month, and year,

The output - valid company, moderate path, 16, october, 2022)

createReservation(int, int, bool, bool): null

<40, 5, True, True, tour.reservationDetails>

tour.reservationDetails = [40, 5, True, True]

createNewEvent(String, String, String): String[ ]

<"Ice Cream Day", "October 13th, 2022", "8:00am", event.eventDetails>

event.eventDetails = ["Taco Day", "April 12th, 2022", "1:00pm"] addTrailDetails(int, varchar): null

<7, "Tanzania National Park", trail.trailDetails>

trail.trailDetails = [7, "Tanzania National Park"]

cancelEvents(): null

event.eventDetails = ["Ice Cream Day", "October 29th, 2022", "12:00pm"]

cancelTours(): null

tour.reservationDetails = [0, 0, False, False]

closeTrails(): null

trail.trailDetails = [0, “Tanzania National Park”]

closePark(): null

park.parkOpeningHours = [0, 0, 0, 0, 0, 0, 0]

park.parkClosingHours = [0, 0, 0, 0, 0, 0, 0]