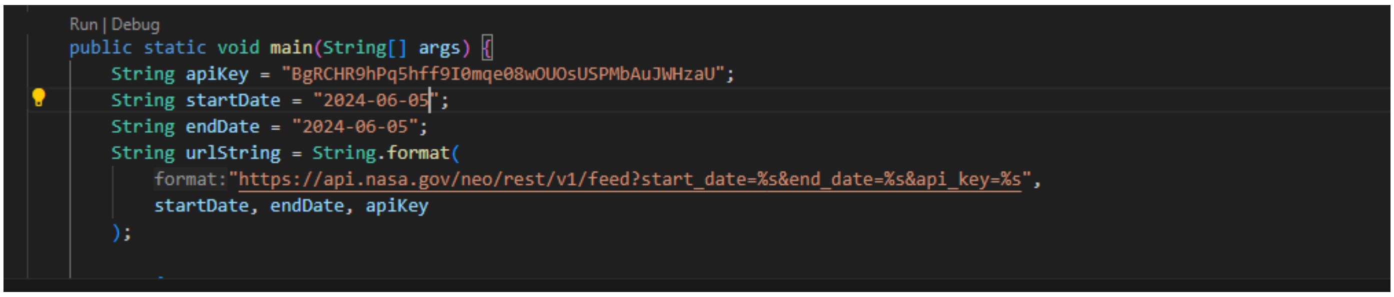
Team Pi
Katerina Cowan
Jonathon Delemos
Legna Saca Archuleta
Mohammad Taufique Imrose
CJ Pallari

CSC 179–01
June 17, 2024

NASA NeoWs API Testing

The scope of this document is to define boundary cases which the API should pass and fail, that is the purpose of API testing. The basis for defining these test cases is to determine if the API has the prescribed boundaries. Our aim was to be certain the data structure, the NASA asteroid repository, had the proper boundary limits. If inaccuracies are detected, we will document them. In this document, our team will declare a strategy to take for boundary analysis API testing. Using mathematics, we will generate equivalence partition cases based on the API input parameters. There are two main parameters which allow our API to function which we will focus on, in addition, there are two lesser complex parameters that allow our API to function. The main parameters of our function are the start date and the end date. We are proud to report we have found an infinite number of combinations that work for all dates. We discovered that leading zeros could be added to any of the data parameters, allowing for input string variations. We can express this results as a regular expression “0*(date)”. These are the dates our program is scanning for asteroids. Listed below are the input parameter types and their position within our program.

First Parameter: Personal NASA API KEY
Second Parameter: YYYY-MM- DD (Start Date)
Third Parameter: YYYY-MM-DD (End Date)
Fourth Parameter: URL, API Key, Start Date, End Date



```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "2024-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
        format:"https://api.nasa.gov/neo/rest/v1/feed?start_date=%s&end_date=%s&api_key=%s",
        startDate, endDate, apiKey
    );
```

Boundary Definitions

For the more complex of our input parameters, the start and end dates, we have devised partitions to test for common integer inputs. For our API, the data will only be printed if the start and end dates are identical. For these parameters, these tests will require 4n, 4n+1, 4n-1, 2n, 2n-1, 2n+1, 2n, 2n +1, 2n -1 cases for each equivalence partition date. As defined in lecture, we will test for each N+1`, N, N-1. In each case we should see an expected response, N+1/N-1 will fail, and N will pass. Because the dates are strict in their input requirements, we will not be testing within a range of N values. For the less complex of our input parameters, our personal NASA API and constructed URL string, we can use the same system. For the API, we will test an N (API), N+1 (API + one alphanumeric), and finally N-1 (API -1 alphanumeric). Then we have the URL. The same system will be followed.

Condition	*Valid eq. classes*	*Invalid eq. Classes*
nr of inputs	1	0, > 1
Input type	integer	integer

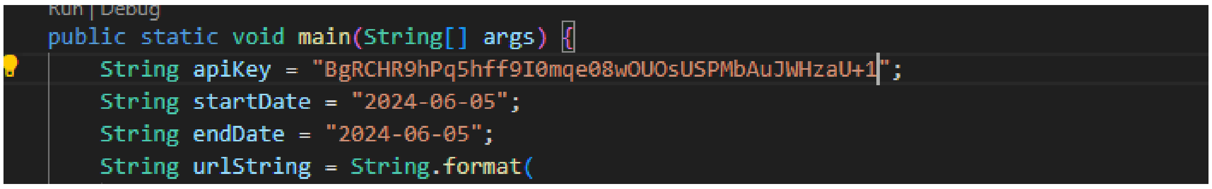
Condition	Valid Partitions	Invalid Partitions	Domain	Input Type
Personal API	1	N<1 / N>1	Infinite	Alphanumeric
Start Date Year	1 (4 numbers)	N<1 / N>1	{0-12}	Integers {0-9}
Start Date Month	1 (2 numbers) “0*Date”	N<1 / N>1	{0-30}	Integers {0-9}
Start Date Day	1 (2 numbers) “0*Date”	N<1 / N>1	{0-30}	Integers {0-9}
End Date Year	1 (4 numbers) “0*Date”	N<1 / N>1	{0-12}	Integers {0-9}
End Date Month	1 (2 numbers) “0*Date”	N<1 / N>1	{0-30}	Integers {0-9}
End Date Day	1 (2 numbers) “0*Date”	N<1 / N>1	{0-30}	Integers {0-9}
API	1 (Correct Combination)	N<1 / N>1	a-Z0-9	Alphanumeric

API Test Cases

In this segment of the document, we explore the boundaries of the invalid equivalence cases. For each parameter, we will exactly define the test cases which should fall within our parameters, but still produce an invalid result. After producing a result, we will display the information for the community to evaluate.

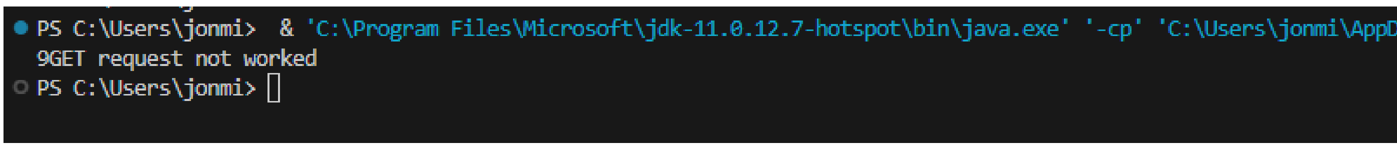
Case 1: Personal KEY

N+1 - Expected Fail



```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU+1";
    String startDate = "2024-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

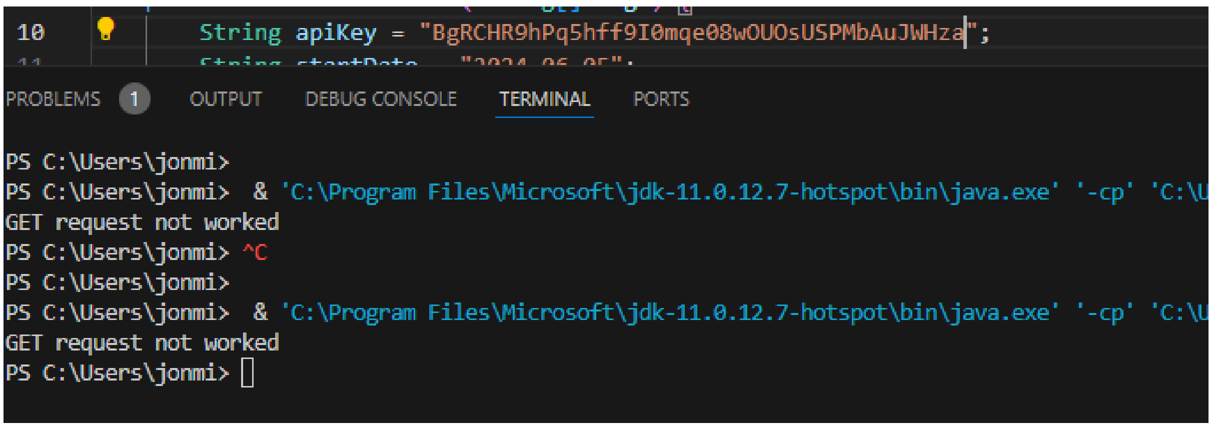
In this test case, I have simply added + 1 to our personal Nasa API key. Now, the program should fail. The result is listed below:



```
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe' '-cp' 'C:\Users\jonmi\AppD
9GET request not worked
PS C:\Users\jonmi> 
```

As expected, the test case failed. This is our N+1 case. Next, we will test our N-1 case. That is removing values from the accepted API key.

N-1 - Expected Fail



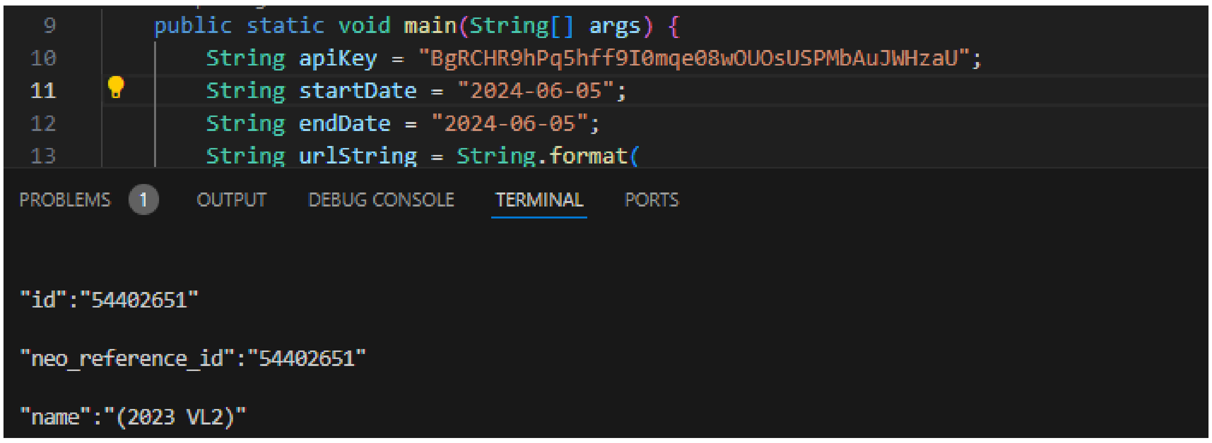
```
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHza";
```

```
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe' '-cp' 'C:\U
GET request not worked
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe' '-cp' 'C:\U
GET request not worked
PS C:\Users\jonmi> 
```

In the above case, it failed. This is due to the correct parameters not being entered for the API Key.

Finally, we will test N to verify it works.

N - Expected Pass



```
    public static void main(String[] args) {
        String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
        String startDate = "2024-06-05";
        String endDate = "2024-06-05";
        String urlString = String.format(
```

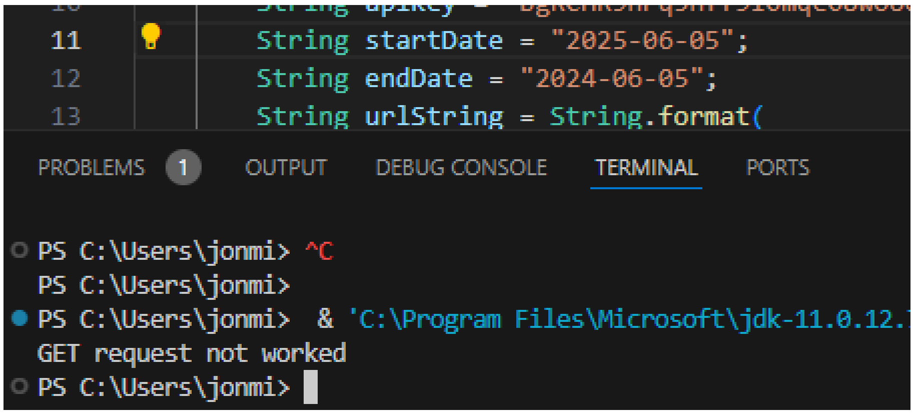
```
"id":"54402651"

"neo_reference_id":"54402651"

"name":"(2023 VL2)"
```

The test case is a success, as expected. The program reports all nearby/orbiting asteroids around earth.

Case 2: Start Date (YYYY, MM, DD)

In this segment of our analysis, we will focus on equivalence partitioning of the valid and invalid classes of the start date. The start date has three components: Year, month, day. For each integer combination, we will create results N+1, N, N-1 to test the validity of our API. First, we will start with a future date

Start Date YYYY (N of future) - Expected Fail



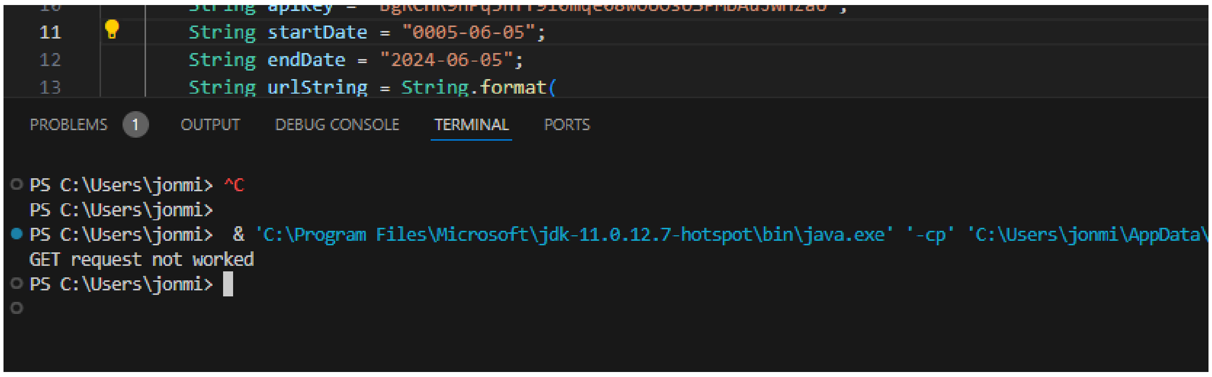
```
String startDate = "2025-06-05";
String endDate = "2024-06-05";
String urlString = String.format(
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.
GET request not worked
PS C:\Users\jonmi>
```

Here, we can see the proper number of integers were entered, but the date has not yet occurred. Therefore, the system cannot retrieve any information. The request fails, as expected.

Start Date YYYY (N of Distant Past) - Expected Fail



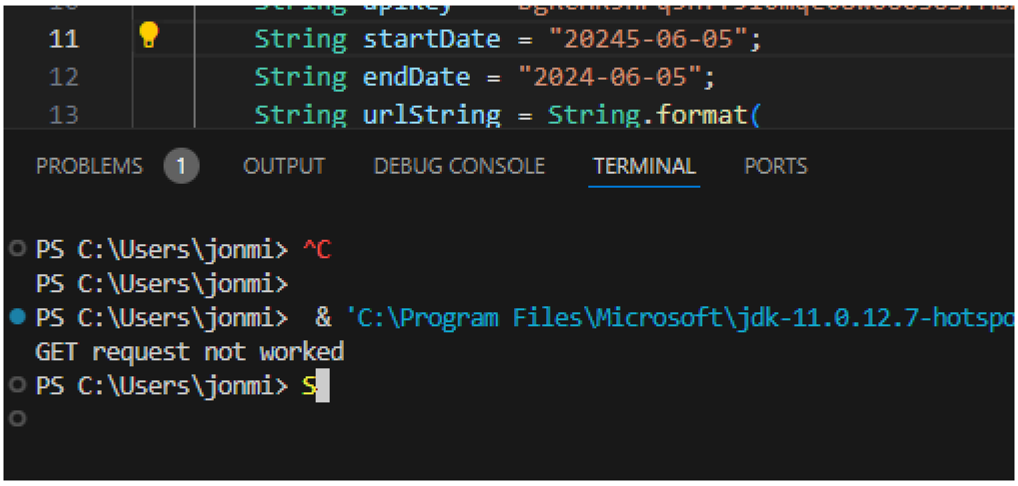
```
String startDate = "0005-06-05";
String endDate = "2024-06-05";
String urlString = String.format(
```

PROBLEMS 1 OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe' '-cp' 'C:\Users\jonmi\AppData\
GET request not worked
PS C:\Users\jonmi>
```

In this test case, we examine the star date range 0005 - 2024, which causes an error due to the lack of valid information present for a large segment of that range.

Start Date YYYY N +1 - Expected Fail

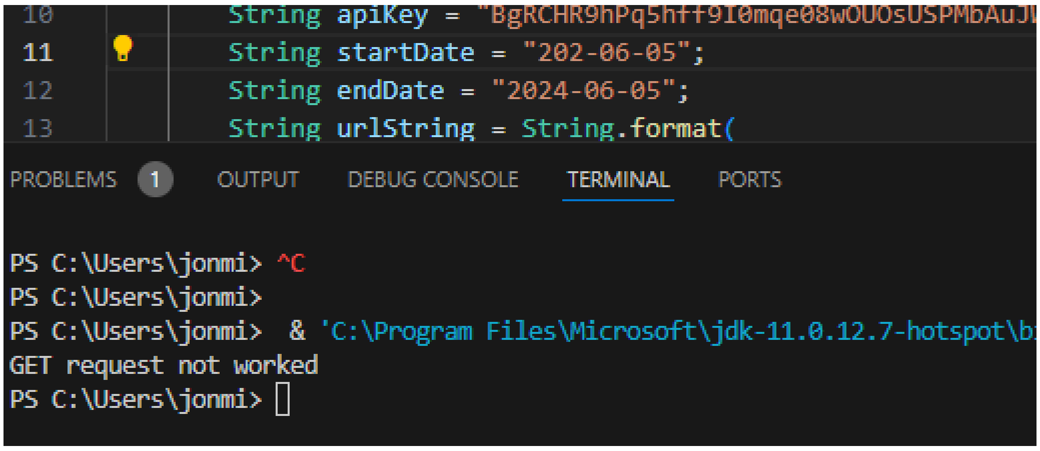


```
        String startDate = "20245-06-05";
        String endDate = "2024-06-05";
        String urlString = String.format(
```

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspo
GET request not worked
PS C:\Users\jonmi> S
```

In this case, we see the addition of an extra integer causes an immediate get request failure. Of course, we cannot retrieve information from an invalid future start date. The result is a failure to get the data. However, the position and specific integer value added affect the answer. If we add zeros in the starting integer position, we see a successful data retrieval.

Start Date YYYY N-1 - Expected Fail

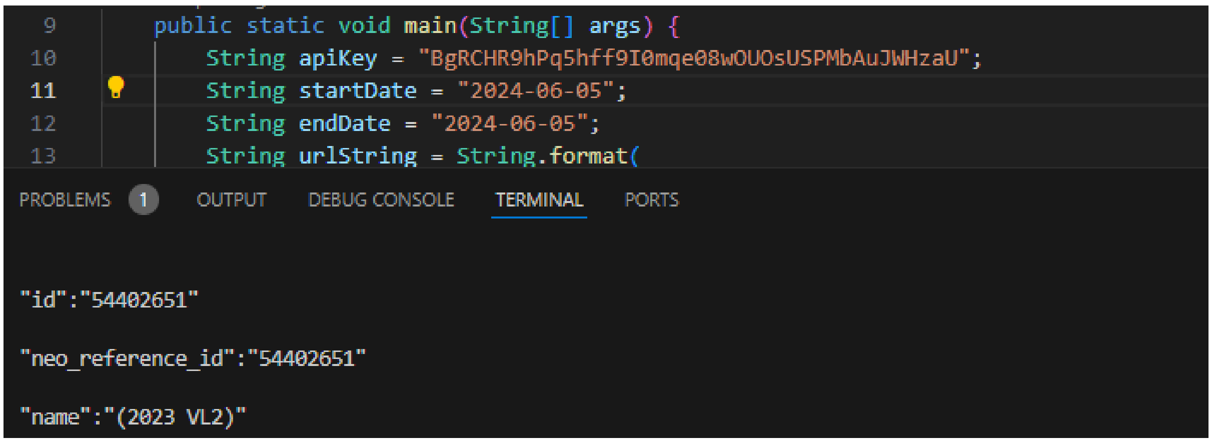


```
        String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJ
        String startDate = "202-06-05";
        String endDate = "2024-06-05";
        String urlString = String.format(
```

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\b
GET request not worked
PS C:\Users\jonmi> 
```

For this test case, we have removed one integer from the start date year column. With this removed integer, the valid format for API calls has been disrupted. The result is a failure.

Start Date YYYY N - Expected Pass

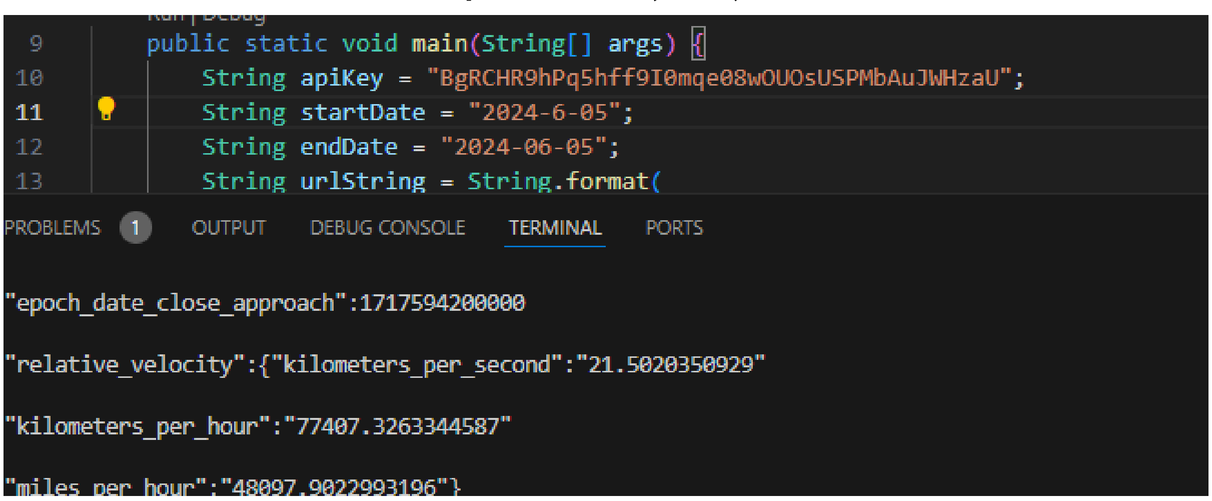


```
    public static void main(String[] args) {
        String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
        String startDate = "2024-06-05";
        String endDate = "2024-06-05";
        String urlString = String.format(
```

```
"id":"54402651"

"neo_reference_id":"54402651"

"name":"(2023 VL2)"
```

The nearby asteroid data is printed. The year segment of the start date has the correct number of integers and falls within the range of dates of collected data. This test case is a success. We have defined the partitions for the start date year.

Start Date MM N-1 - Expected Fail (Pass)

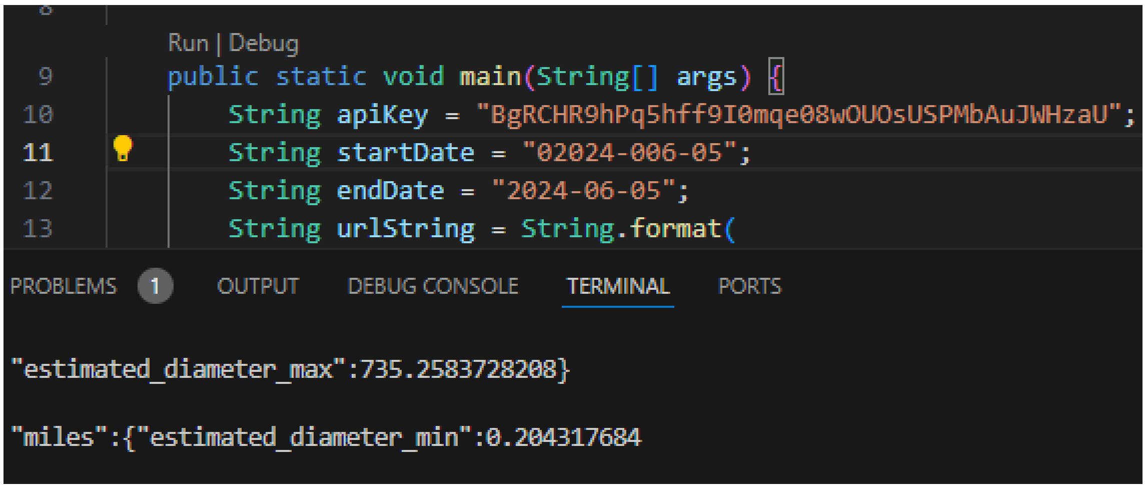


```
    public static void main(String[] args) {
        String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
        String startDate = "2024-6-05";
        String endDate = "2024-06-05";
        String urlString = String.format(
```

```
"epoch_date_close_approach":1717594200000

"relative_velocity":{"kilometers_per_second":"21.5020350929"

"kilometers_per_hour":"77407.3263344587"

"miles_per_hour":"48097.9022993196"}
```

The nearby asteroid data is printed! The month segment of the start date has the incorrect number of integers yet falls within the range of dates of collected data. This test case is a success. Our API is capable of interpreting a single integer as a month parameter. This is remarkable and noted within our documentation. It is now suspected all months and day parameters can be accepted as single integers as well.

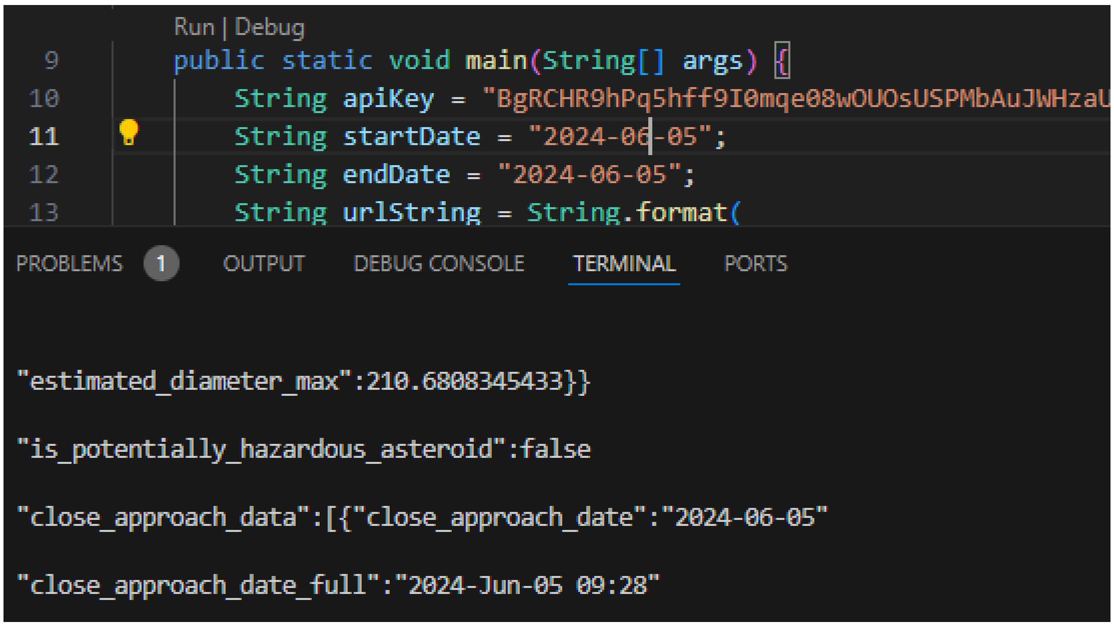
Start Date MM N+1 - Expected Fail


```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "02024-006-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
"estimated_diameter_max":735.2583728208}

"miles":{"estimated_diameter_min":0.204317684
```

Adding a zero to the month column does not create an error, thus, we can create infinite strings of inputs for the year, month and day columns. As we can see, I have added zeros to both the year and month start date parameters. This has not affected the answer.

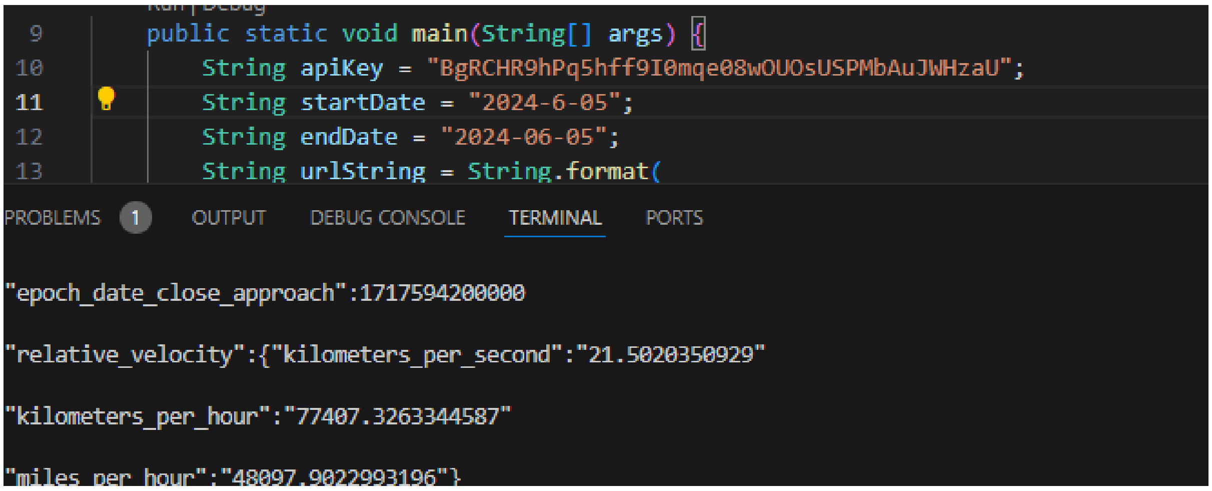
Start Date MM N - Expected Pass


```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU
    String startDate = "2024-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
"estimated_diameter_max":210.6808345433}}

"is_potentially_hazardous_asteroid":false

"close_approach_data":[{"close_approach_date":"2024-06-05"

"close_approach_date_full":"2024-Jun-05 09:28"
```

In this test case, we find the data is printed. The correct parameters are entered and the nearby asteroid data is retrieved. It should be noted that throughout the testing we discovered that N-1 is also included in the accepted set for month and day parameters, given the second integer is not zero.

Start Date DD N-1 - Expected Fail

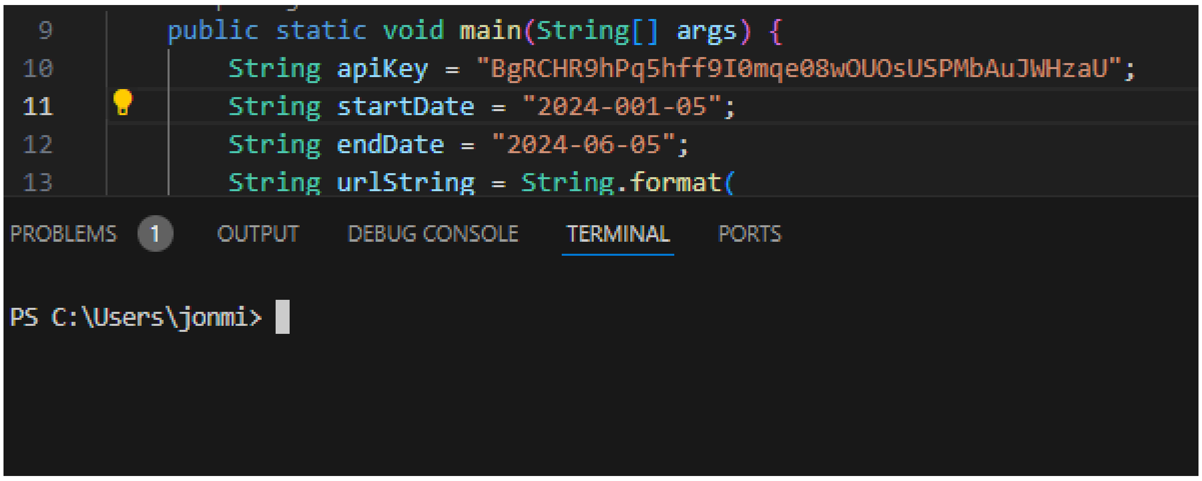


```
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "2024-6-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
"epoch_date_close_approach":1717594200000

"relative_velocity":{"kilometers_per_second":"21.5020350929"

"kilometers_per_hour":"77407.3263344587"

"miles_per_hour":"48097.9022993196"}
```

Again, we find the asteroid data is printed! This is quite remarkable, it seems we don't need to include a leading integer of 0, therefore our parameters change for single digit entries.

Start Date DD N+1 - Expected Fail

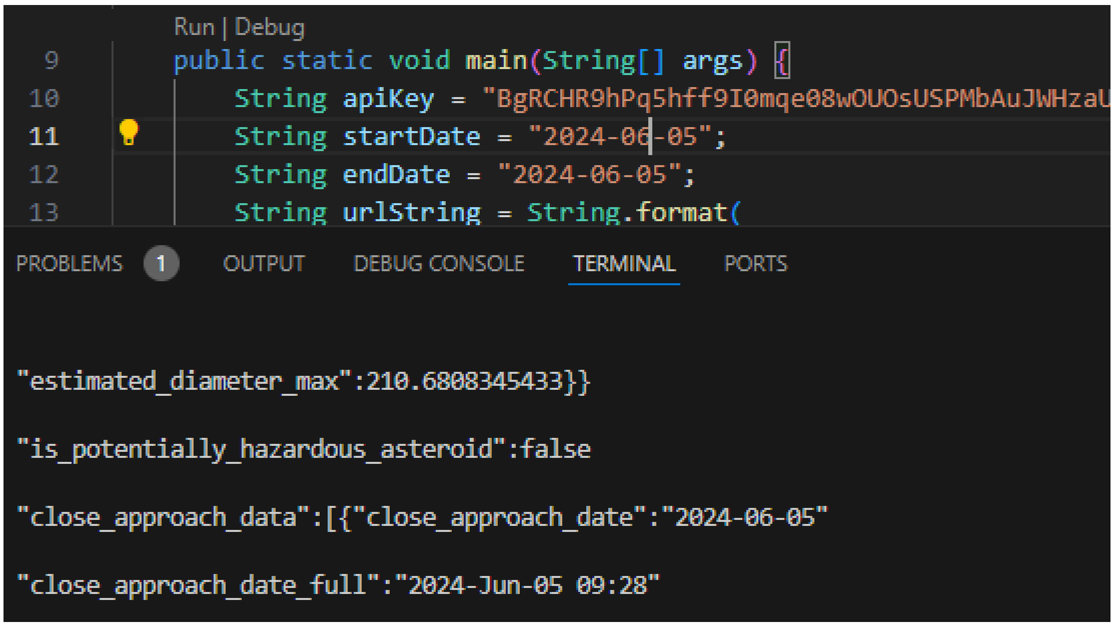


```
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "2024-001-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
PS C:\Users\jonmi>
```

Adding an extra number results in a retrieval error. What is odd is the program can interpret a single integer, but cannot detect multiple leading zeros which create the same value. It is very interesting that this API is coded in such a way.

Start Date DD N - Expected Pass



```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU
    String startDate = "2024-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

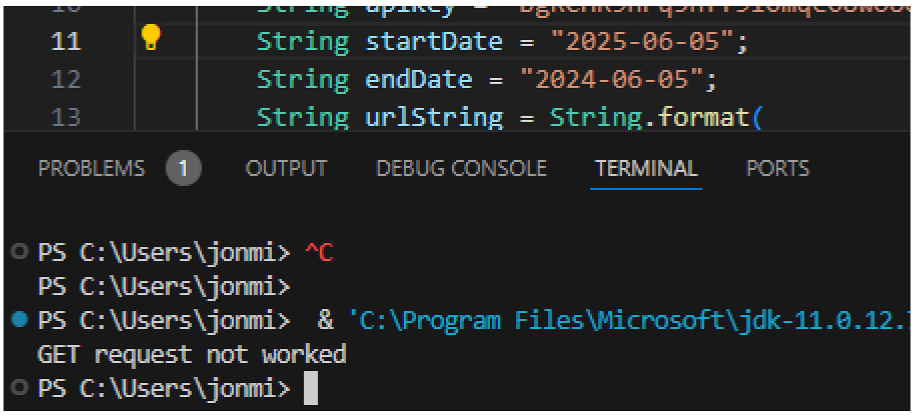
```
"estimated_diameter_max":210.6808345433}}

"is_potentially_hazardous_asteroid":false

"close_approach_data":[{"close_approach_date":"2024-06-05"

"close_approach_date_full":"2024-Jun-05 09:28"
```

In this test case, we find the data is printed. The date is properly formatted and the API is successful in retrieving its information. The nearby asteroid data is printed and the test case is a success.

Case 3: End Date (YYYY, MM, DD)

For this portion of our API testing document, we will focus on the various fields that are needed to satisfy an end date input. In Case 3, we have three different components: year, month, and day.

End Date YYYY (N of future) - Expected Fail



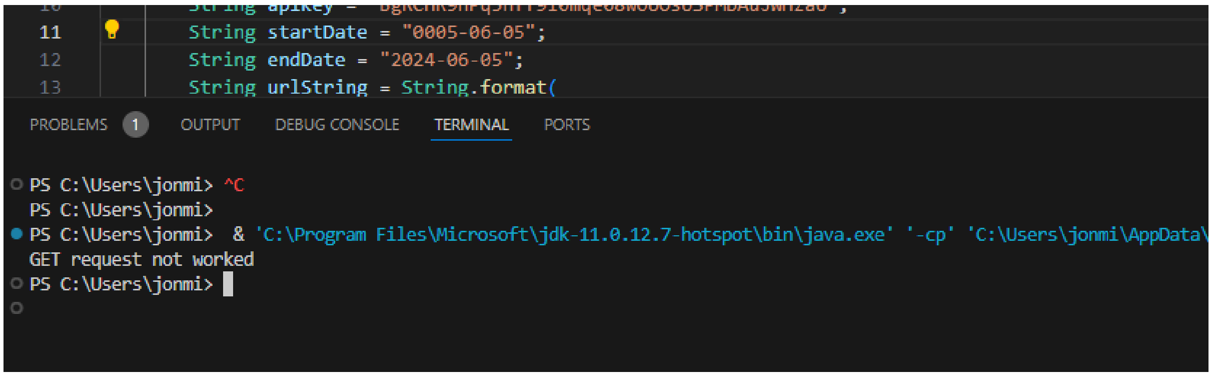
```
    String startDate = "2025-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.
GET request not worked
PS C:\Users\jonmi>
```

By inputting a future date which has yet to happen, we see the system fail to retrieve the data from the NASA repositories. In addition, it should be noted that the API cannot withdraw information from multiple days. The days must synchronized to allow for the API to function properly.

End Date YYYY (N of Distant Past) - Expected Fail

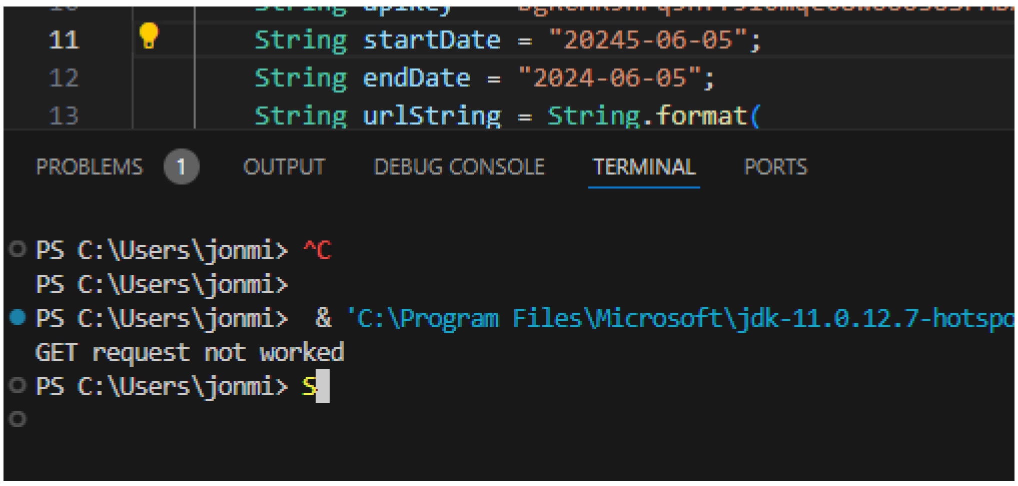


```
String startDate = "0005-06-05";
String endDate = "2024-06-05";
String urlString = String.format(
```

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe' '-cp' 'C:\Users\jonmi\AppData\
GET request not worked
PS C:\Users\jonmi>
```

In this test case, we see the API fail to retrieve the data for reasons previously specified. There is no data present within the database during the dates that are present.

End Date YYYY N +1 - Expected Fail

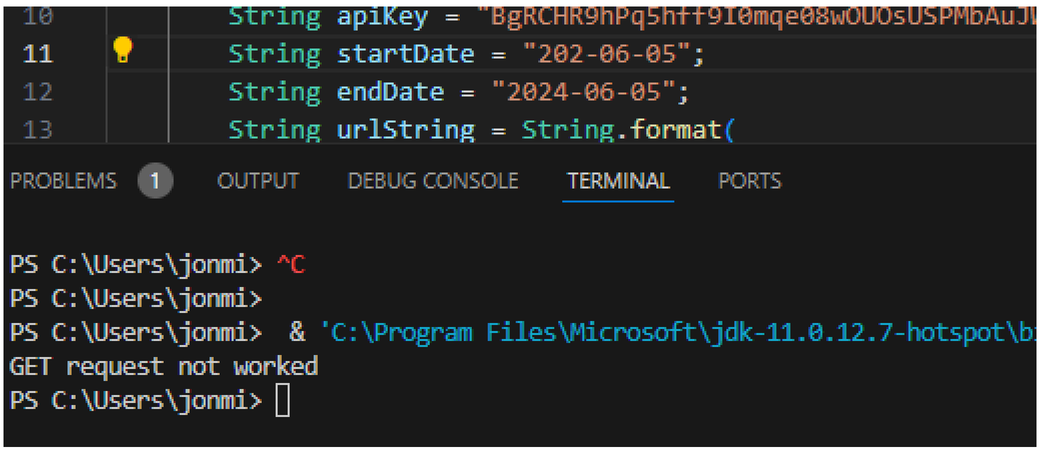


```
String startDate = "20245-06-05";
String endDate = "2024-06-05";
String urlString = String.format(
```

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspo
GET request not worked
PS C:\Users\jonmi> S
```

In this case, we see the addition of an extra integer causes an immediate get request failure. Of course, we cannot retrieve information from an invalid future start date. The result is a failure to get the data. There is no possible way to retrieve data from the future, therefore the system returns a GET request failure message.

End Date YYYY N-1 - Expected Fail

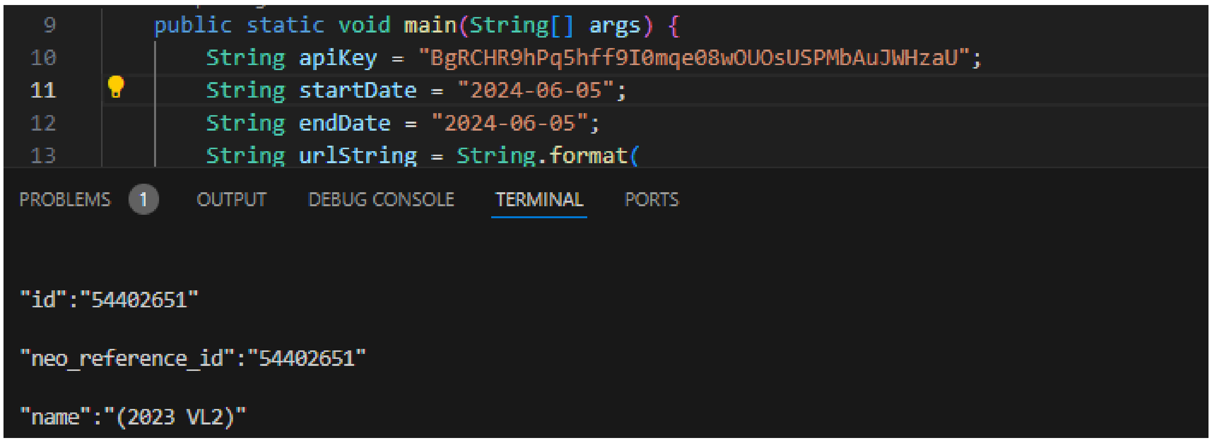


```
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJ
    String startDate = "202-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\b
GET request not worked
PS C:\Users\jonmi> 
```

For this test case, we observe the end date String has one less integer than required for the API to function. Of course, there is also a range of 4 digit integers for which the API can retrieve information. We cannot withdraw information before the database was established.

End Date YYYY N - Expected Pass

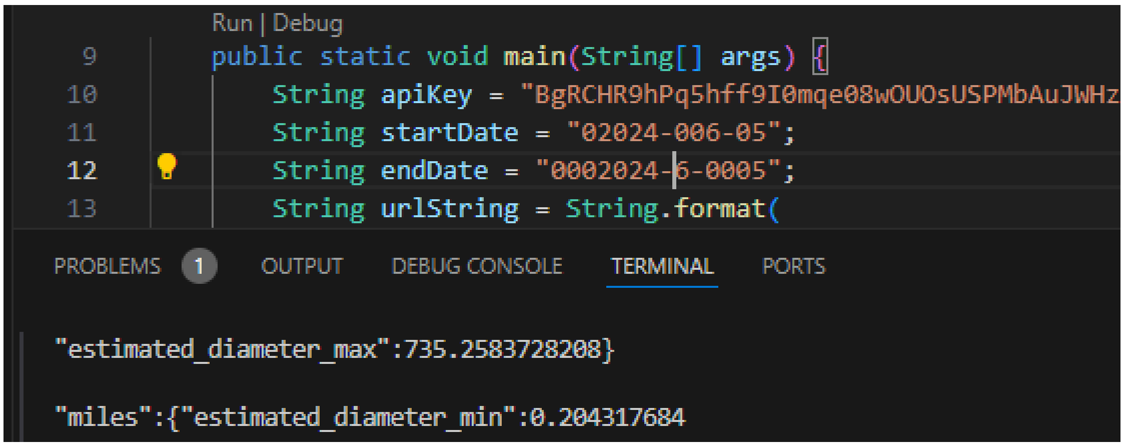


```
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "2024-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
"id":"54402651"

"neo_reference_id":"54402651"

"name":"(2023 VL2)"
```

The nearby asteroid data is printed. The impact of the year being entered correctly is visibly registered, we can see a large array of information displayed. Again, it is important to note that the repository can only report identical startDate and endDate requests.

End Date MM N-1 - Expected Fail



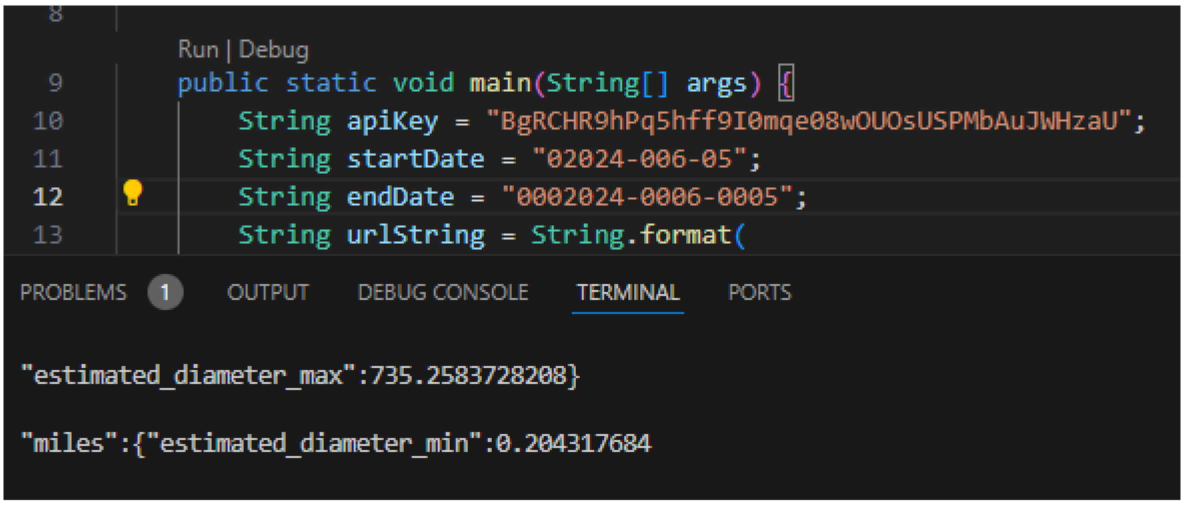
```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHz
    String startDate = "02024-006-05";
    String endDate = "0002024-6-0005";
    String urlString = String.format(
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
"estimated_diameter_max":735.2583728208}

"miles":{"estimated_diameter_min":0.204317684
```

The nearby asteroid data is printed! Again, with the endDate functioning exactly like the start date, we can remove a leading 0 integer from the date and still experience a success in retrieval. This will not work for integers with an ending zero, or adding additional leading zeros to the input for month.

End Date MM N+1 - Expected Fail (Pass/Fail)



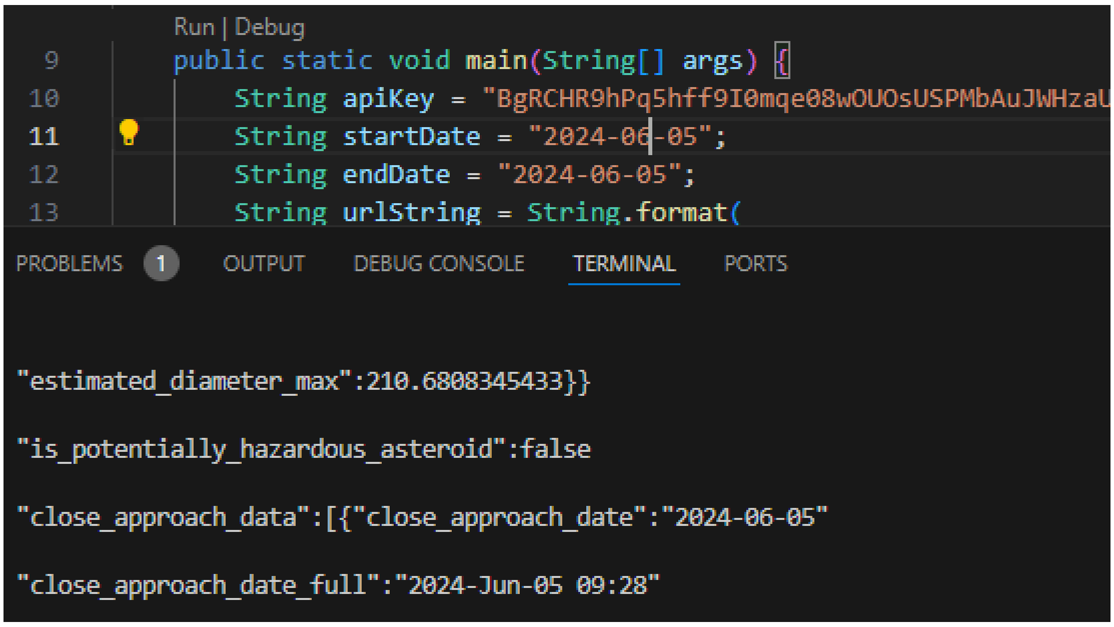
```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "02024-006-05";
    String endDate = "0002024-0006-0005";
    String urlString = String.format(
```

PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

```
"estimated_diameter_max":735.2583728208}

"miles":{"estimated_diameter_min":0.204317684
```

It seems the inverse is applicable here! Remarbarkable! Adding an extra zero which should be interpreted by humans as still 01 does not create a retrieval error. What is odd is the program can interpret a single integer and can detect multiple leading zeros which create the same value for start date. At any rate, it was expected to fail and it did not! Therefore, any number of zero digits can be entered within this parameter, assuming all leading digits are zeros prior to the trailing final two digits.

End Date MM N - Expected Pass

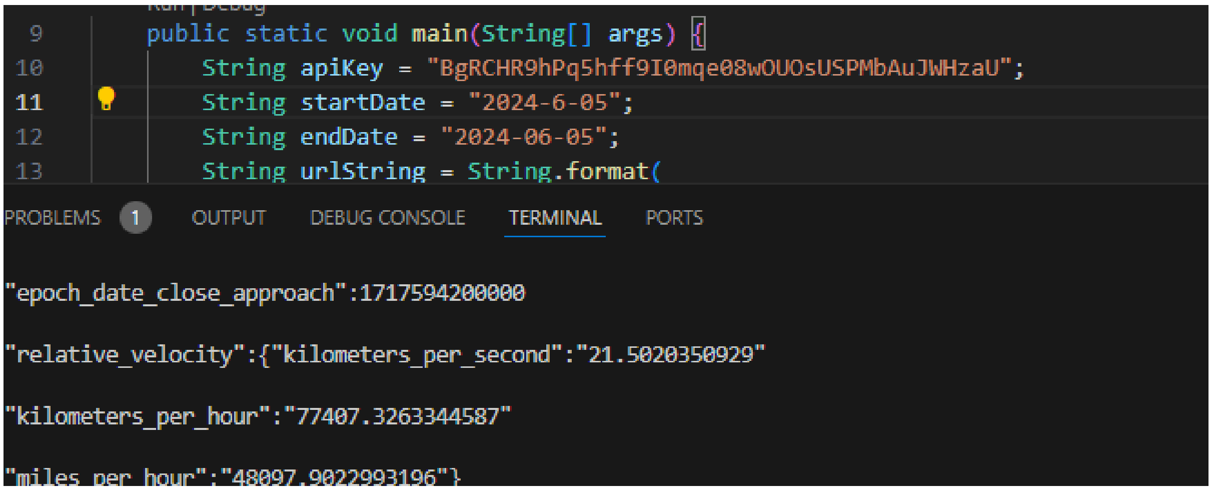


```
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU
    String startDate = "2024-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
"estimated_diameter_max":210.6808345433}}

"is_potentially_hazardous_asteroid":false

"close_approach_data":[{"close_approach_date":"2024-06-05"

"close_approach_date_full":"2024-Jun-05 09:28"
```

In this test case, we find the data is printed. The correct parameters are entered and the nearby asteroid data is retrieved. It should be noted that throughout the testing we discovered that N-1 is also included in the accepted set for month and day parameters, given the second integer is not zero.

End Date DD N-1 - Expected Fail (Pass)

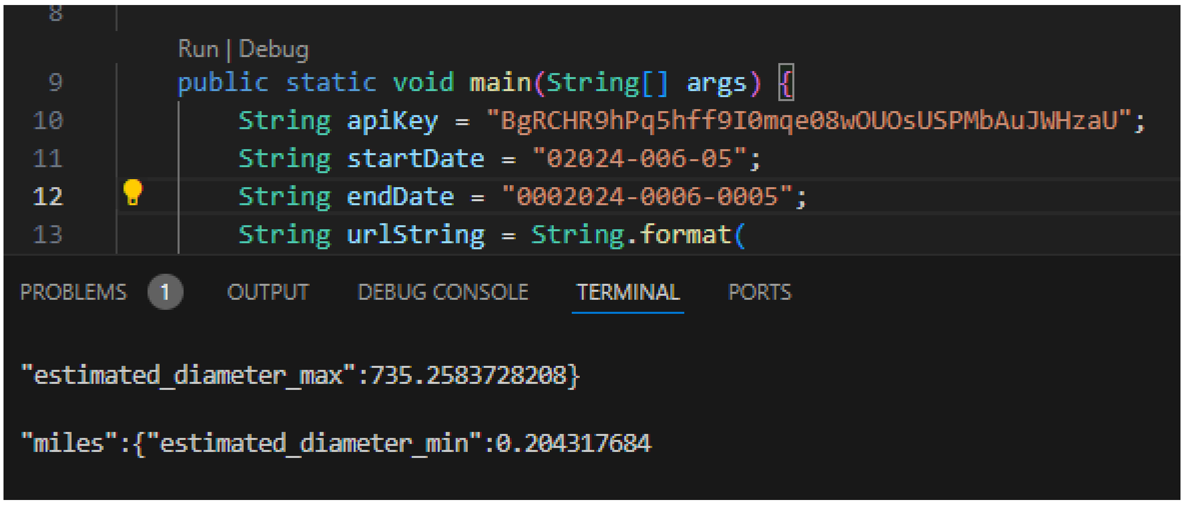


```
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "2024-6-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
"epoch_date_close_approach":1717594200000

"relative_velocity":{"kilometers_per_second":"21.5020350929"

"kilometers_per_hour":"77407.3263344587"

"miles_per_hour":"48097.9022993196"}
```

Again, we find the asteroid data is printed! This is quite remarkable, it seems we don't need to include a leading integer of 0, therefore our parameters change for single digit entries.

End Date DD N+1 - Expected Fail (Pass/Fail)

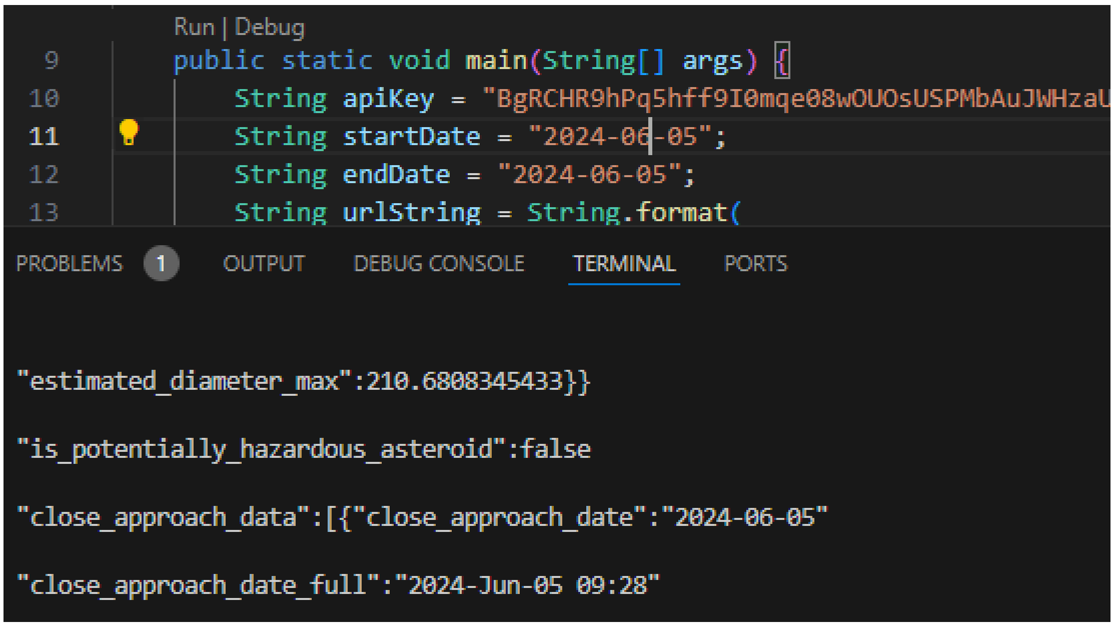


```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU";
    String startDate = "02024-006-05";
    String endDate = "0002024-0006-0005";
    String urlString = String.format(
```

```
"estimated_diameter_max":735.2583728208}

"miles":{"estimated_diameter_min":0.204317684
```

Testing our theory, we see a random number of zeros added to chosen parameter positions. The end date has multiple zeros leading in several parameter positions.

End Date DD N - Expected Pass



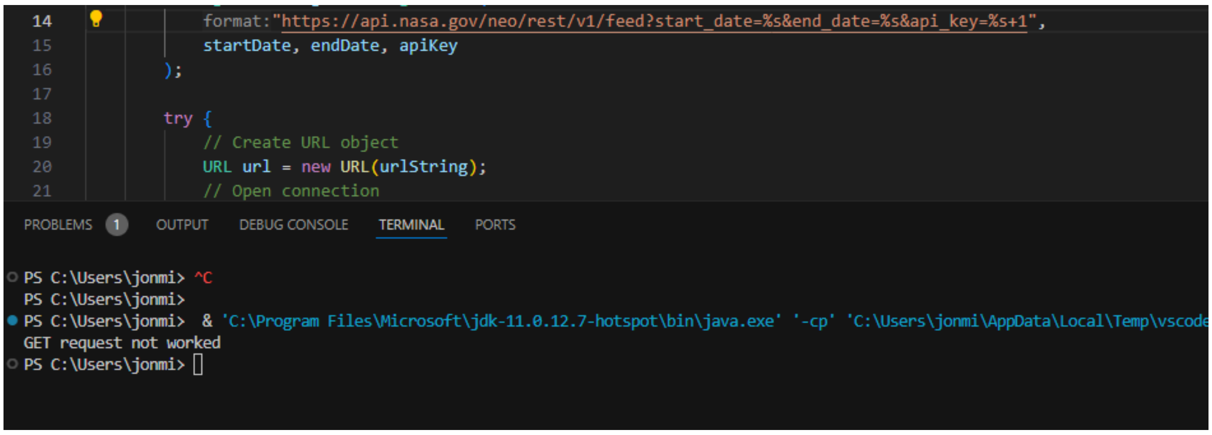
```
Run | Debug
public static void main(String[] args) {
    String apiKey = "BgRCHR9hPq5hff9I0mqe08wOUOsUSPMbAuJWHzaU
    String startDate = "2024-06-05";
    String endDate = "2024-06-05";
    String urlString = String.format(
```

```
"estimated_diameter_max":210.6808345433}}

"is_potentially_hazardous_asteroid":false

"close_approach_data":[{"close_approach_date":"2024-06-05"

"close_approach_date_full":"2024-Jun-05 09:28"
```

In this test case, we find the data is printed. The date is properly formatted and the API is successful in retrieving its information. The nearby asteroid data is printed and the test case is a success.

Case 4: API Key

API N+1 - Expected Fail

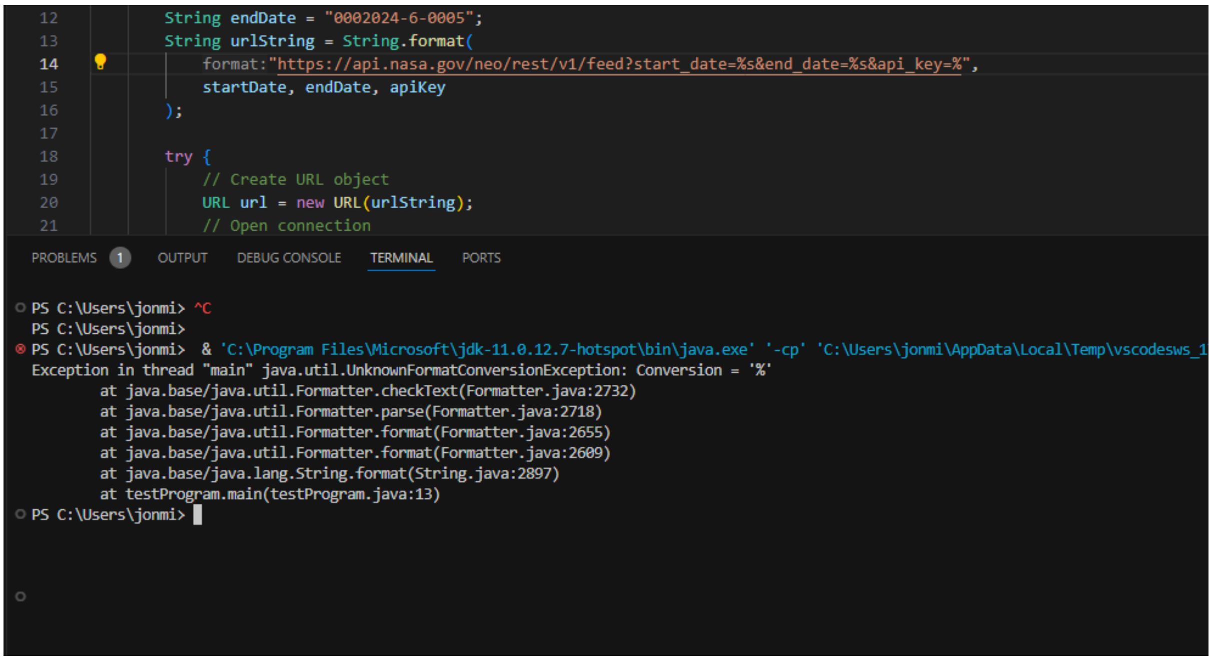


```
            format:"https://api.nasa.gov/neo/rest/v1/feed?start_date=%s&end_date=%s&api_key=%s+1",
            startDate, endDate, apiKey
        );

        try {
            // Create URL object
            URL url = new URL(urlString);
            // Open connection
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe' '-cp' 'C:\Users\jonmi\AppData\Local\Temp\vscode
GET request not worked
PS C:\Users\jonmi> 
```

In this case, we have added a one to the API key value. As a result, the GET request failed. This was to be expected. The API URL is a link we are trying to create. With an incorrect reach for the material, we cannot GET anything. The location we reached for was nonexistent.

API N-1 - Expected Fail

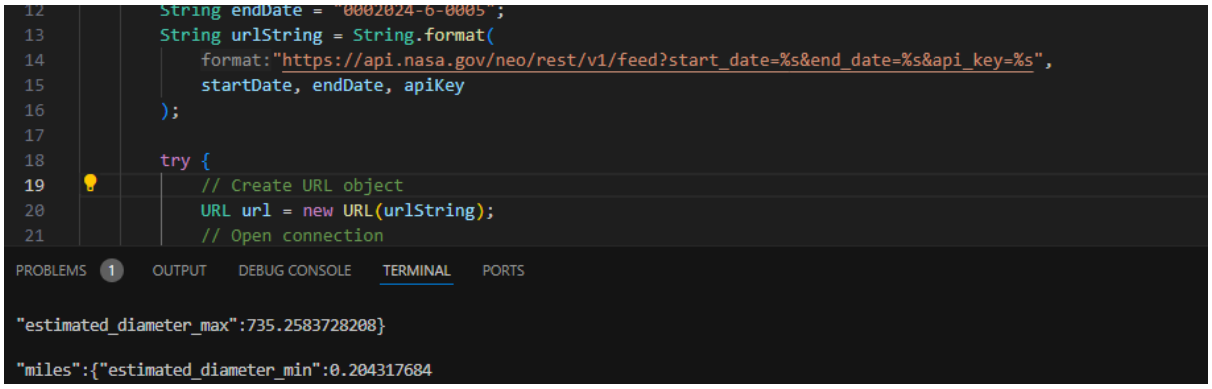


```
        String endDate = "0002024-6-0005";
        String urlString = String.format(
            format:"https://api.nasa.gov/neo/rest/v1/feed?start_date=%s&end_date=%s&api_key=%",
            startDate, endDate, apiKey
        );

        try {
            // Create URL object
            URL url = new URL(urlString);
            // Open connection
```

```
PROBLEMS 1   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS C:\Users\jonmi> ^C
PS C:\Users\jonmi>
PS C:\Users\jonmi>  & 'C:\Program Files\Microsoft\jdk-11.0.12.7-hotspot\bin\java.exe' '-cp' 'C:\Users\jonmi\AppData\Local\Temp\vscodesws_1
Exception in thread "main" java.util.UnknownFormatConversionException: Conversion = '%'
        at java.base/java.util.Formatter.checkText(Formatter.java:2732)
        at java.base/java.util.Formatter.parse(Formatter.java:2718)
        at java.base/java.util.Formatter.format(Formatter.java:2655)
        at java.base/java.util.Formatter.format(Formatter.java:2609)
        at java.base/java.lang.String.format(String.java:2897)
        at testProgram.main(testProgram.java:13)
PS C:\Users\jonmi> 
```

In this case, if I remove the string formatting, which is to say remove the API key value, we see a java.util.UnknownFormatConversion exception. This means that the compiler cannot interpret our request without a valid apiKey parameter to be formatted within the string. As a result, this had failed.

API N - Expected Pass



```
        String endDate = "0002024-6-0005";
        String urlString = String.format(
            format:"https://api.nasa.gov/neo/rest/v1/feed?start_date=%s&end_date=%s&api_key=%s",
            startDate, endDate, apiKey
        );

        try {
            // Create URL object
            URL url = new URL(urlString);
            // Open connection
```

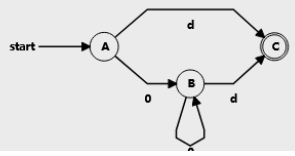
```
"estimated_diameter_max":735.2583728208}

"miles":{"estimated_diameter_min":0.204317684
```

Here, we see the data is successfully printed. This was to be expected, for we have formatted the string apiKey correctly.

Conclusions

For the purpose of this API testing, we identified valid partitions for the API. In testing, we were able to find infinite string variations of acceptable partitions. As an example, we will display a DFA to express our findings. (0 is 0, d = date)

NFA STATE	DFA STATE	TYPE
{0,1,3}	A	
{1,2,3}	B	
{4}	C	accept



In conclusion, we were successful in defining the boundaries of this API and its potential edge cases.