Gebze Technical University
Computer Engineering

CSE 222
2017 Spring

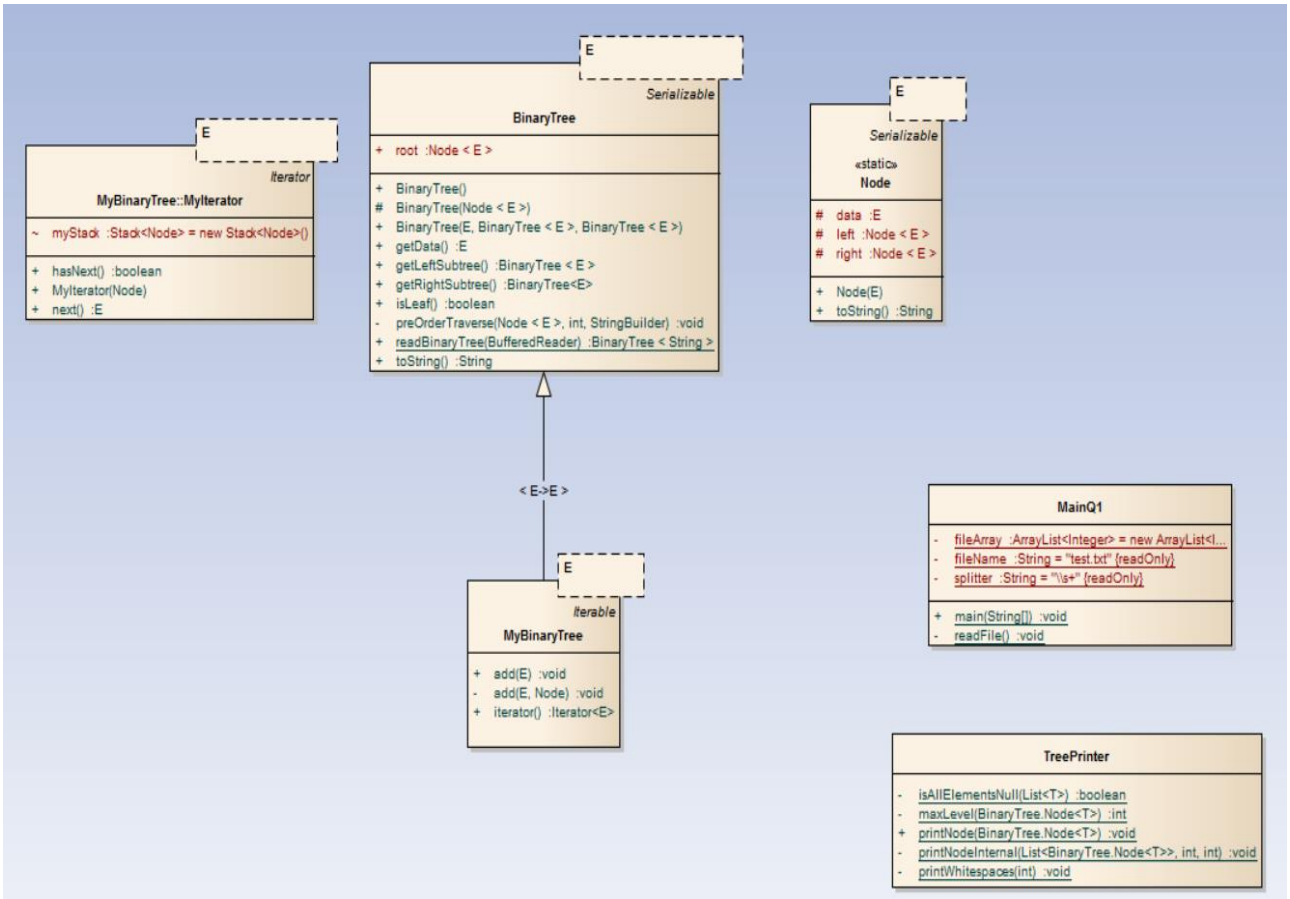
HOMEWORK 05 REPORT

Elif Şeyma ARMAĞAN
151044042

Course Assistant:
Nur Banu ALBAYRAK

Q1.1

1. Class Diagram



2. Problem Solutions Approach

Öncelikle MyBinaryTree classımı oluşturmam lazımdı. Bunun için kitabımızda yer alan KWBinaryTree class ını extend ettim. Oluşturduğum class ın içinde kendi iterator class ımı yazdım. İterator un pre-order şekilde traversal yapabilmesi için stack veri yapısını kullandım. Next metodu her bir child a geldiğinde onun önce right sonra left child ını stack e ekledim böylelikle pre-order treversal ın istediği gibi stack'in LIFO şeklinde çalışması nedeniyle önce sol ağaca sonra sağ ağaca erişebildim.

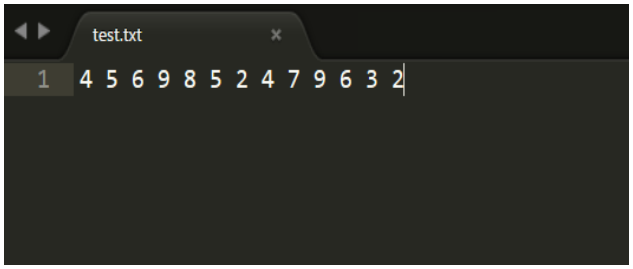
Ağacın görünümünün kodu test eden kişi icin daha anlaşılır olabilmesi için kullandığım TreePrinter isimli class ı http://stackoverflow.com/questions/4965335/how-to-print-binary-tree-diagram sayfasından aldım.

Binary tree ye eleman ekleme işleminde bir kural belirtilmediği için okuduğum test.txt dosyasından elde ettiğim veriler sonucu elemanların random olarak eklendiği bir binary tree oluşturdum. Bu add işlemini MyBinaryTree classında yaptım. 'test.txt' dosyasının tek bir satırdan oluştuğunu varsaydım.

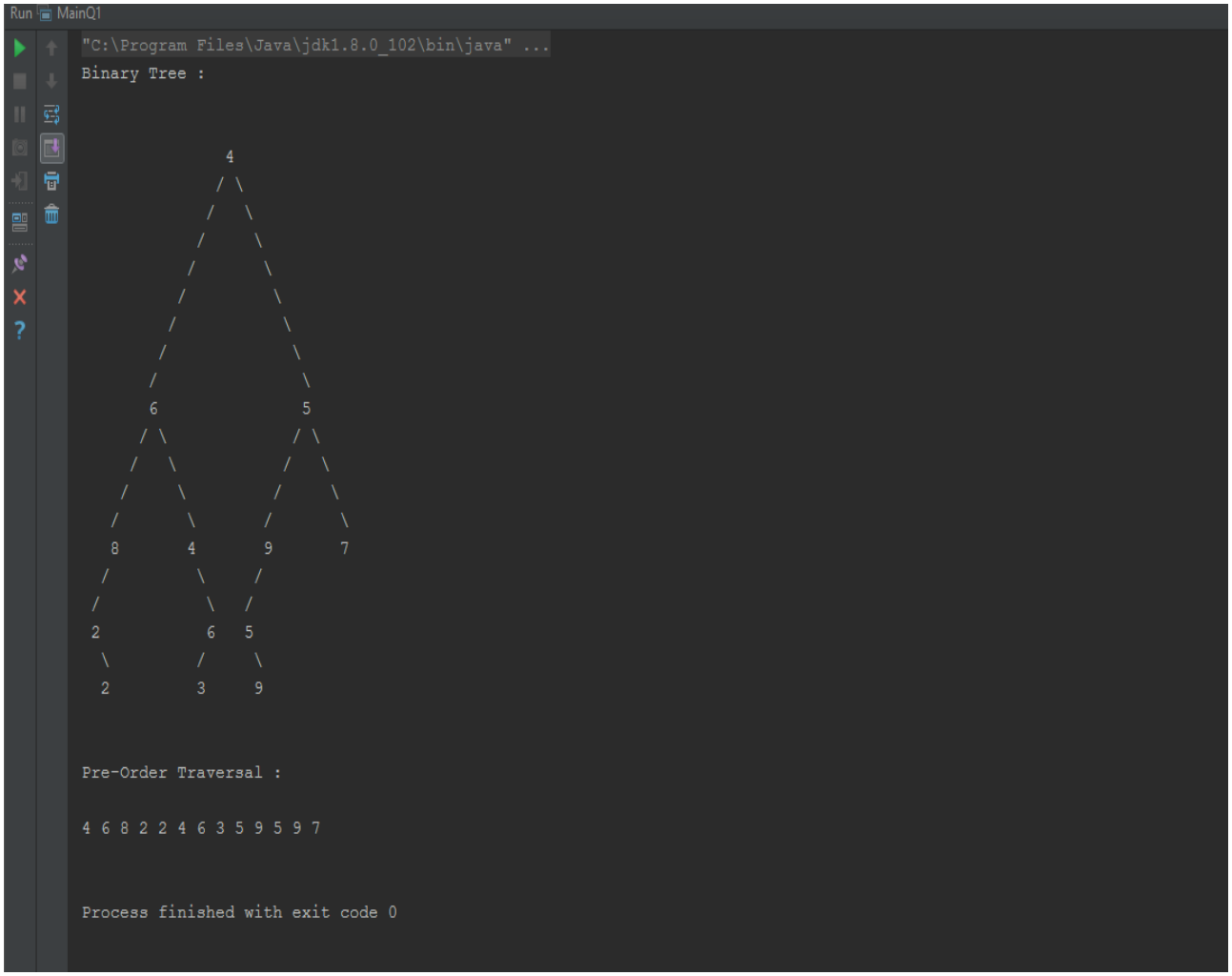
Bu partın test edilebilmesi için MainQ1_1 isimli class ı yazdım.

3. Running and Results

Ağaç oluşturmak için kullandığım test dosyası bu şekildedir.



Run 1:



Run 2:

```
"C:\Program Files\Java\jdk1.8.0_102\bin\java" ...
Binary Tree :
```

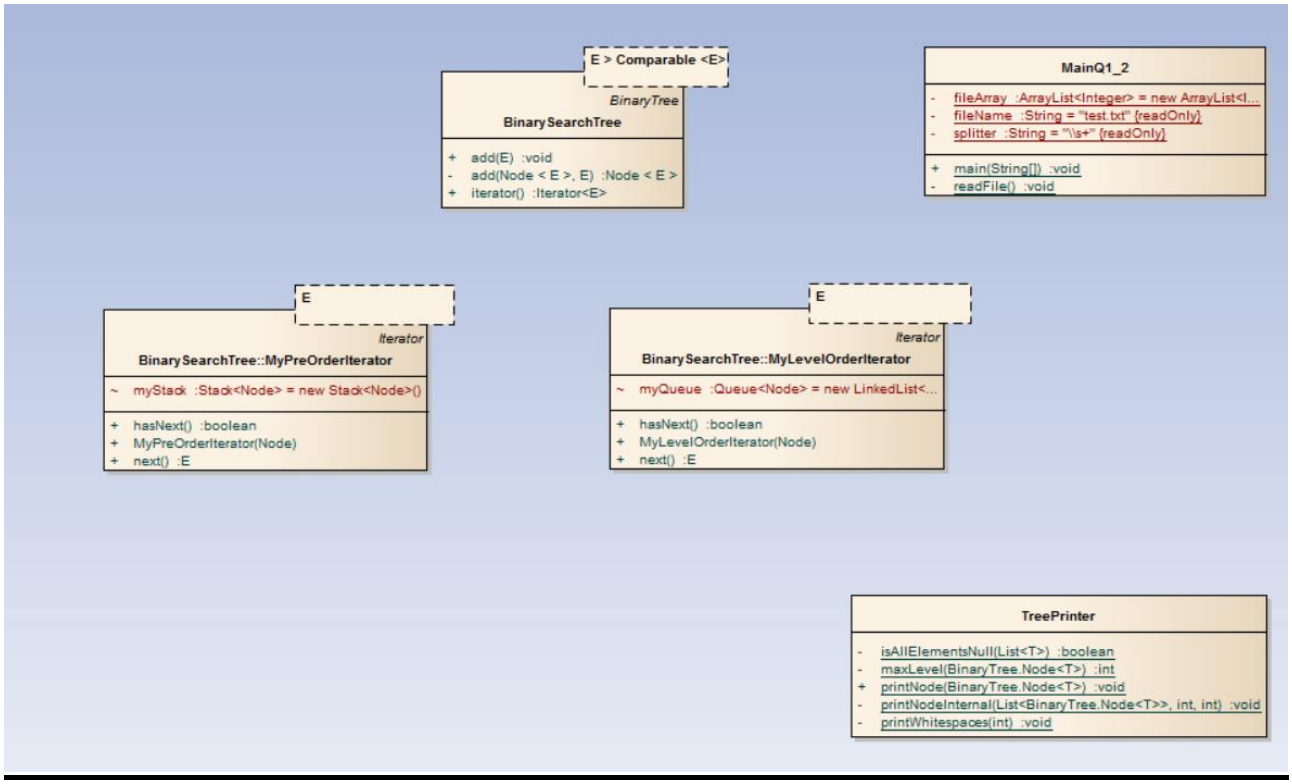

```
Pre-Order Traversal :

4 5 2 4 7 2 6 9 9 3 5 8 6


Process finished with exit code 0
```

Q1.2

1.Class Diagram



2. Problem Solutions Approach

BinarySearchTree class i içinde add metodu ve itearator yazdım. Add metodunu BinarySearchTree insertion kuralında olduğu gibi roottan küçük olan nodeları ağacın sol tarafına, rootan dan büyük olanları ise node un sağ tarafına ekleyerek yaptım.

BinarySearchTree classimin icinde hem level-order traversal yapan iterator hem de pre-order traversal yapan 2 tane iteratörü de yazdım. Test ederken sadece level-order traversal ı test ettim.

İteratör ün level-order şekilde traversal yapabilmesi için queue veri yapısını kullandım. Bu yapıyı kullanmamın nedeni FIFO prensibi ile çalışmasıydı. İteratörün next() metodunu yazarken öncelikle constructor da root node umu queue ya ekledim. Daha sonra next() metodunda queue mdan öncelikle bir eleman çıkardım. Bu çıkardığım elemanın sağ ve sol çocukları varsa onları sırayla queue ya ekledim. Next() metodunu her çağırdığımda bu işlemi tekrarlayarak ağacın tamamını level level queya ekleyerek FIFO prensibi yardımı ile yaptım.

Ağacın görünümünün kodu test eden kişi icin daha anlaşılır olması için kullandığım TreePrinter isimli class ı http://stackoverflow.com/questions/4965335/how-to-print-binary-tree-diagram sayfasından aldım.

Ağaçları oluştururken kullandığım ‘test.txt’ dosyasının tek bir satırdan oluştuğunu varsaydım.

Bu partın test edilebilmesi için MainQ1_2 isimli class ı yazdım.

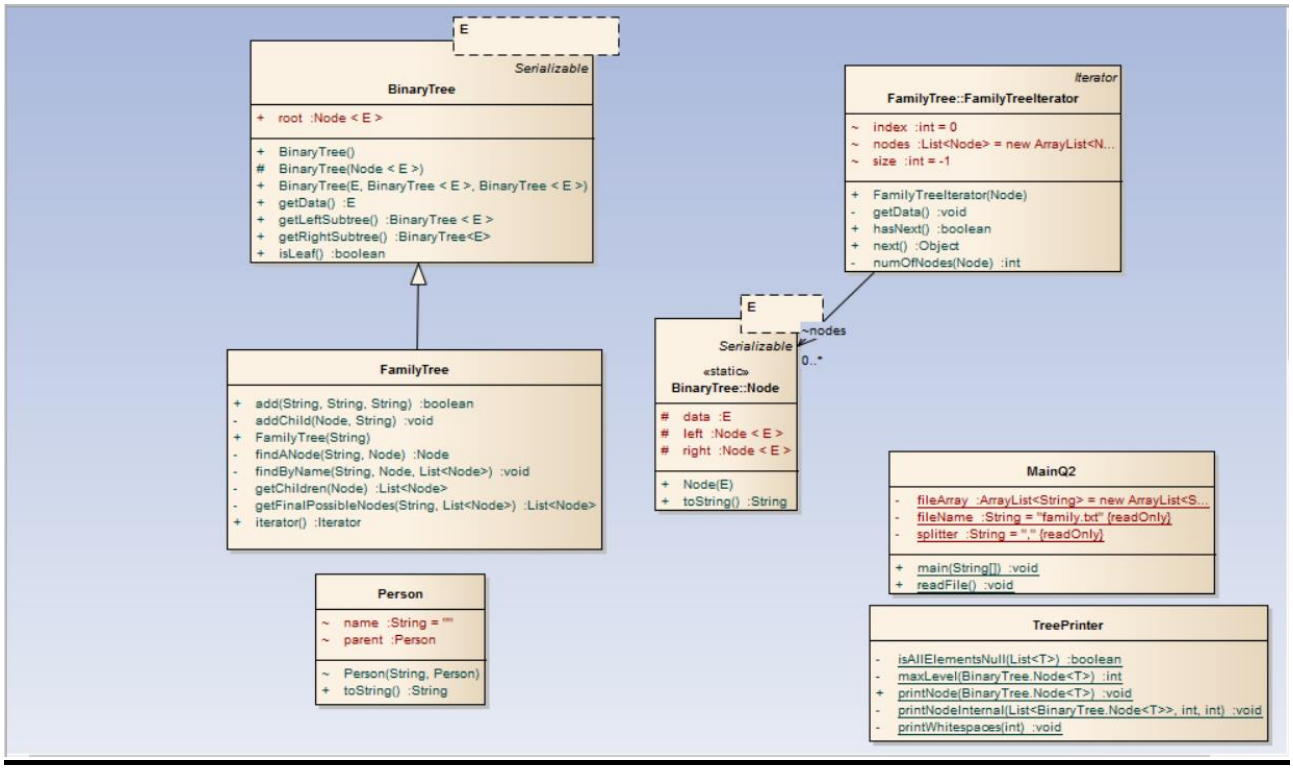
3. Running and Results

Ağaç oluşturmak için kullandığım test dosyası bu şekildedir.

Run :

Q2

1.Class Diagram



2. Problem Solutions Approach

FamilyTree deki nodeları temsil etmesi için inner bir person class ı oluşturdum. Bu class kişinin ismini ve parent ının ismini taşır. Böylelikle nick name ile ilgili işler daha kolay hale gelir.

Family tree ye bir eleman eklemek için ilk önce eşleşen parent ın bulunması lazımdır.Bunun için verilen parent name ine sahip olan tüm nodelar bulunur. Eğer tek bir tane varsa direk eklenir. Eğer hiç yoksa exception fırlatılır. Eğer birden fazla varsa bu kez parent ın nick name ine bakılır. Bulunan tüm node lar için nickname eşleştirmesi yapılır. Eğer tek bir sonuca ulaşırsak bulduğumuz node a ekleriz. Ancak birden fazla varsa eklemek mümkün değildir ve exception fırlatılır.

Eklenecek node bulunduktan sonra bu node un sol tarafına eklememiz gerekmektedir. Eğer left child null ise direkt buraya ekleriz. Null değilse bu left child dan sağa doğru null bulana kadar ilerleyip oraya ekleriz.

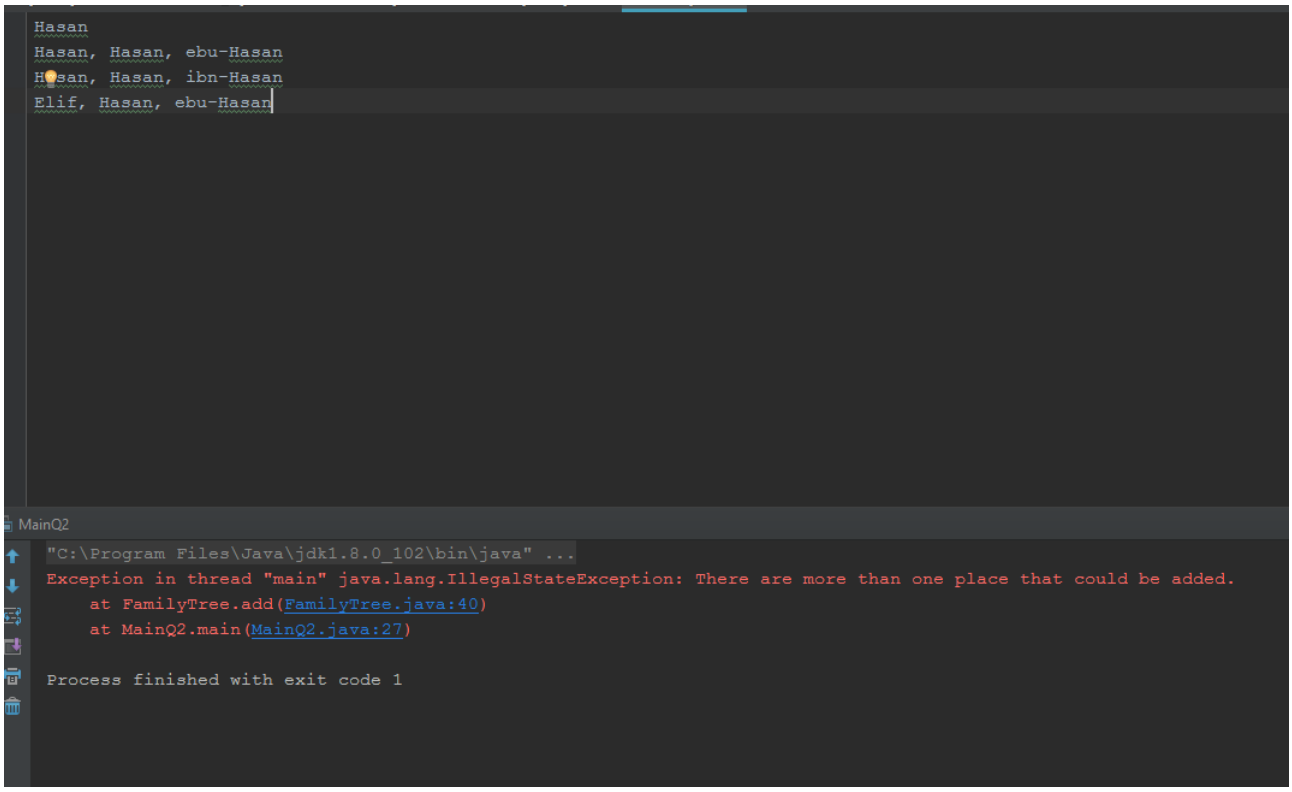
FamilyTree nin iteratöründe ise root tan başlanarak her iterasyonda geçerli node un tüm çocukları bir listeye eklenir. Liste sürekli güncellendiği için tüm nodeların bütün çocukları incelenmiş ve listeye eklenmiş olur. Next() metodunda bu listedeki elemanlar sırayla verilir.

Ağacın görünümünün kodu test eden kişi icin daha anlaşılır olması için kullandığım TreePrinter isimli class ı http://stackoverflow.com/questions/4965335/how-to-print-binary-tree-diagram sayfasından aldım

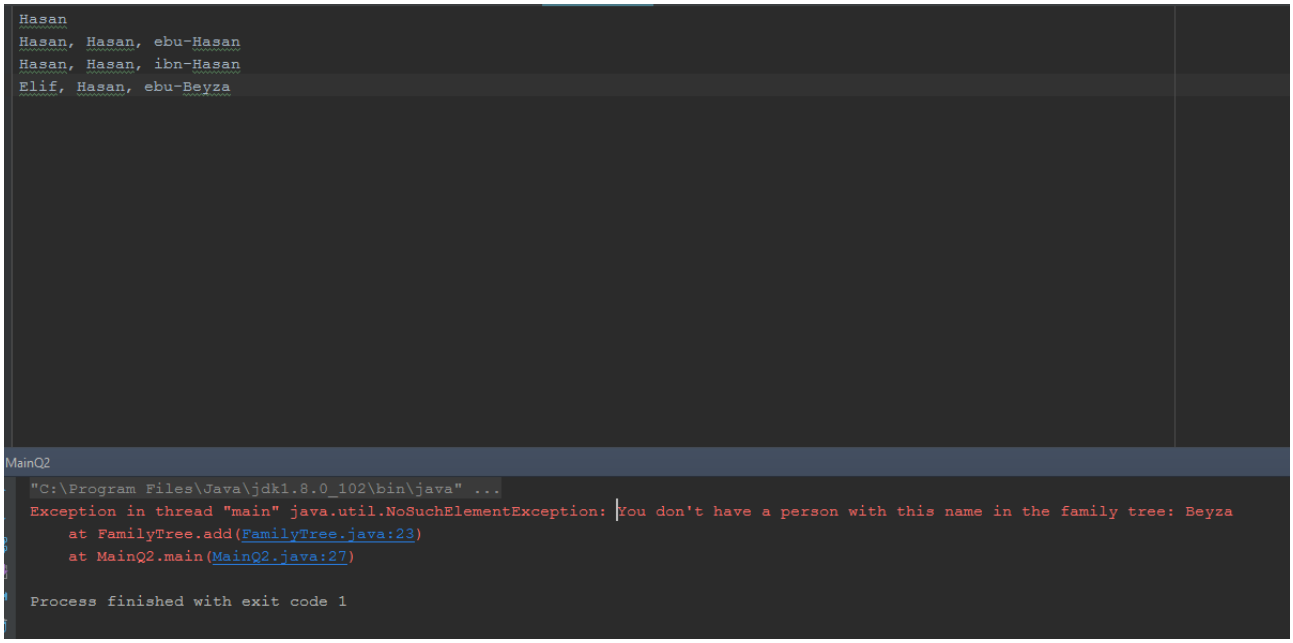
Bu kısmı test etmek için MainQ2 isimli class i yazdım.

3. Running and Results

Eklenecek çocuğun parenti ile aynı isimde 2 kişi varsa exception fırlatılır.



Ağaçta hiç olmayan biri kullanılırsa exception fırlatılır.



Sorunsuz çalışması durumu