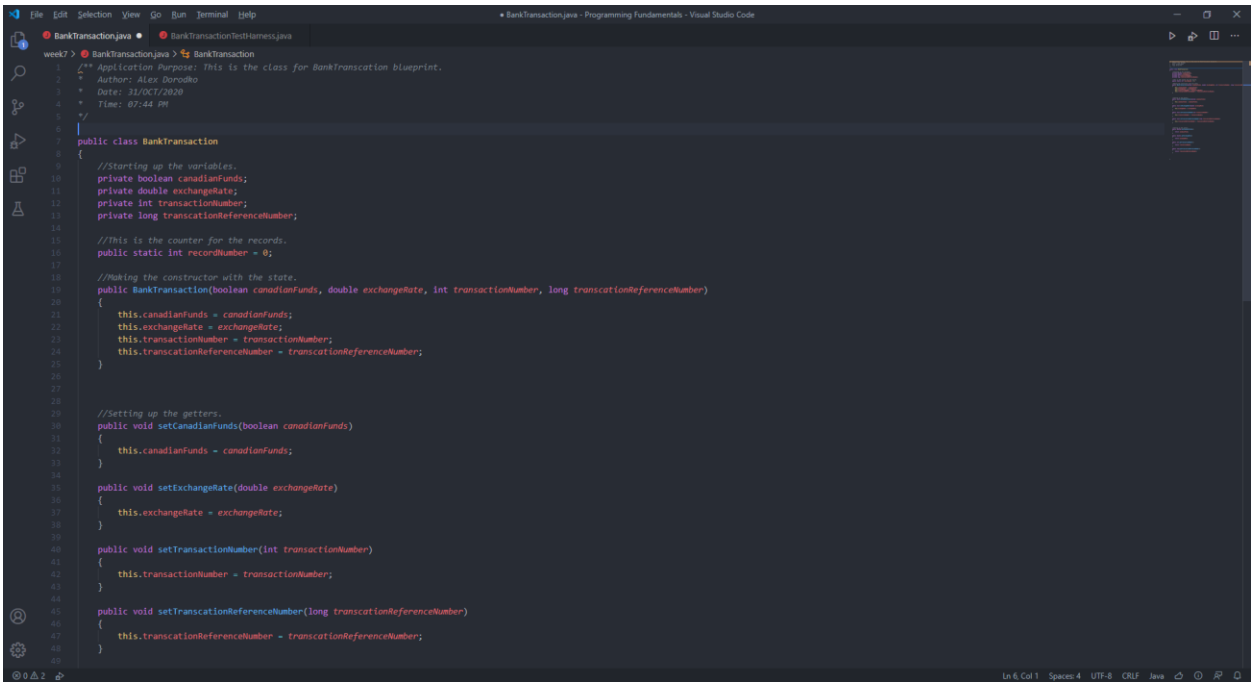
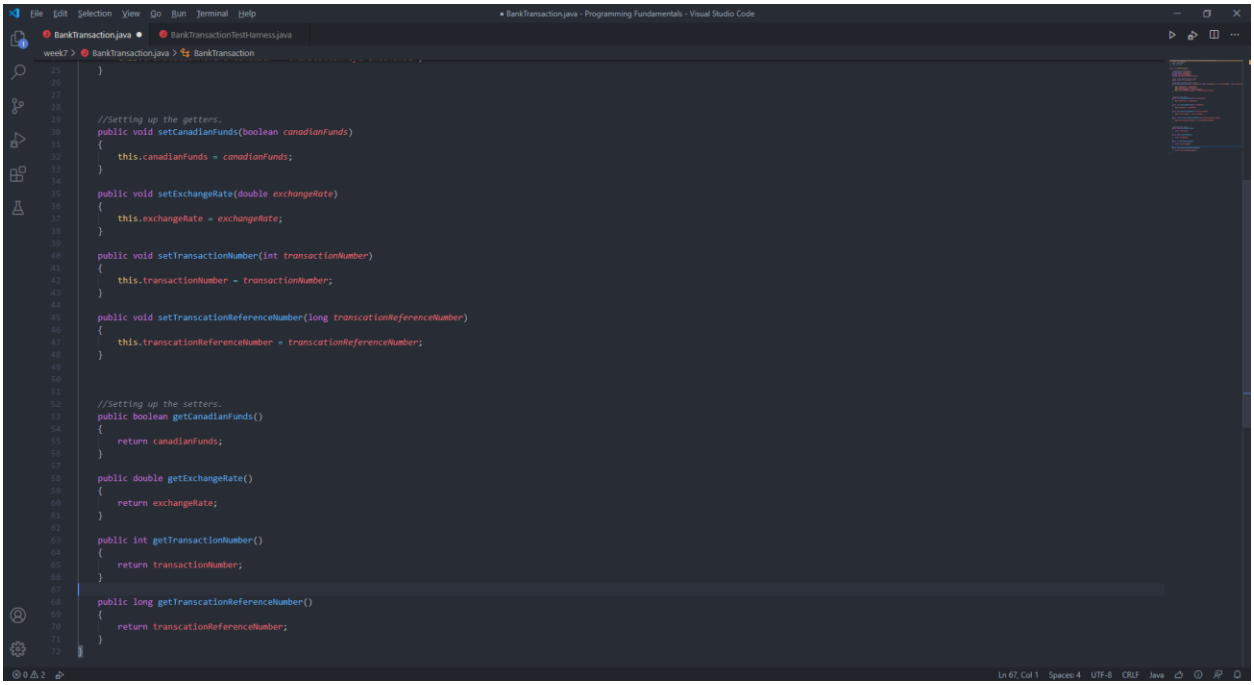
BankTransaction.java

Part 1



```java
/** Application Purpose: This is the class for BankTranscation blueprint.
 *  Author: Alex Dorodko
 *  Date: 31/OCT/2020
 *  Time: 07:44 PM
 */

public class BankTransaction
{
    //Starting up the variables.
    private boolean canadianFunds;
    private double exchangeRate;
    private int transactionNumber;
    private long transcationReferenceNumber;

    //This is the counter for the records.
    public static int recordNumber = 0;

    //Making the constructor with the state.
    public BankTransaction(boolean canadianFunds, double exchangeRate, int transactionNumber, long transcationReferenceNumber)
    {
        this.canadianFunds = canadianFunds;
        this.exchangeRate = exchangeRate;
        this.transactionNumber = transactionNumber;
        this.transcationReferenceNumber = transcationReferenceNumber;
    }



    //Setting up the getters.
    public void setCanadianFunds(boolean canadianFunds)
    {
        this.canadianFunds = canadianFunds;
    }

    public void setExchangeRate(double exchangeRate)
    {
        this.exchangeRate = exchangeRate;
    }

    public void setTransactionNumber(int transactionNumber)
    {
        this.transactionNumber = transactionNumber;
    }

    public void setTranscationReferenceNumber(long transcationReferenceNumber)
    {
        this.transcationReferenceNumber = transcationReferenceNumber;
    }
```

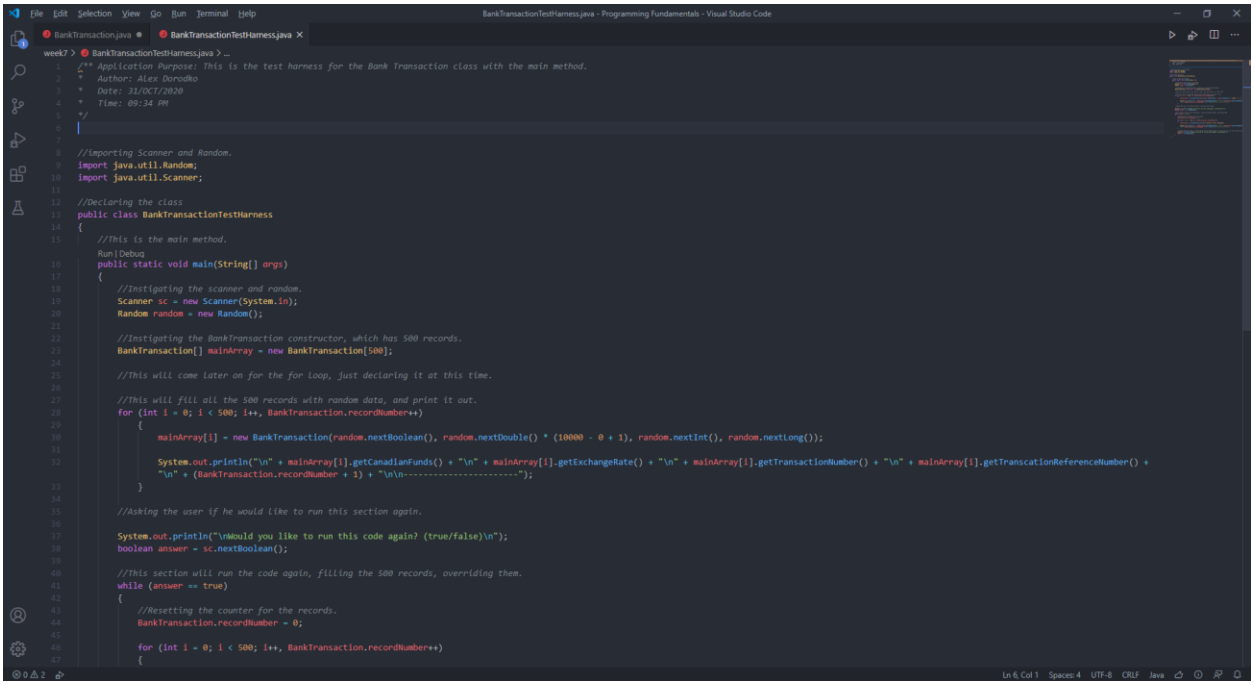
Part 2



```java
    }



    //Setting up the getters.
    public void setCanadianFunds(boolean canadianFunds)
    {
        this.canadianFunds = canadianFunds;
    }

    public void setExchangeRate(double exchangeRate)
    {
        this.exchangeRate = exchangeRate;
    }

    public void setTransactionNumber(int transactionNumber)
    {
        this.transactionNumber = transactionNumber;
    }

    public void setTranscationReferenceNumber(long transcationReferenceNumber)
    {
        this.transcationReferenceNumber = transcationReferenceNumber;
    }



    //Setting up the setters.
    public boolean getCanadianFunds()
    {
        return canadianFunds;
    }

    public double getExchangeRate()
    {
        return exchangeRate;
    }

    public int getTransactionNumber()
    {
        return transactionNumber;
    }

    public long getTranscationReferenceNumber()
    {
        return transcationReferenceNumber;
    }
}
```

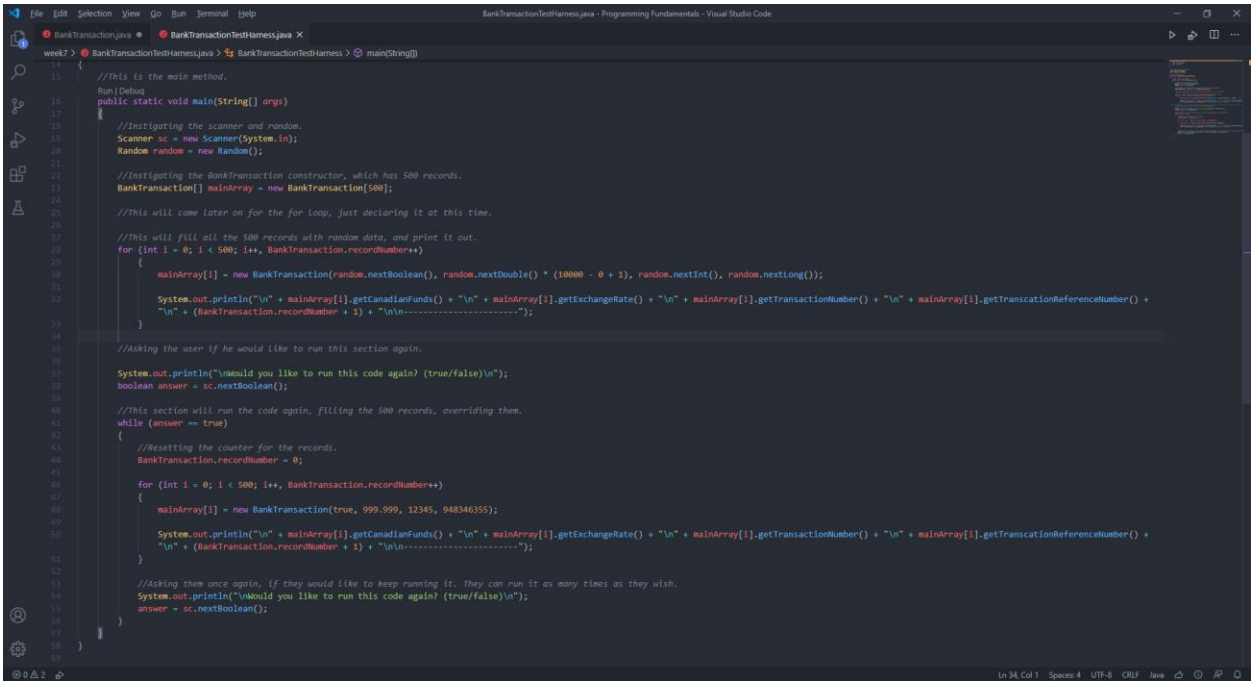
BankTransactionTestHarness.java

Part 1



```java
/** Application Purpose: This is the test harness for the Bank Transaction class with the main method.
 *  Author: Alex Dorodko
 *  Date: 31/OCT/2020
 *  Time: 09:34 PM
 */

//Importing Scanner and Random.
import java.util.Random;
import java.util.Scanner;

//Declaring the class
public class BankTransactionTestHarness
{
    //This is the main method.
    Run | Debug
    public static void main(String[] args)
    {
        //Instigating the scanner and random.
        Scanner sc = new Scanner(System.in);
        Random random = new Random();

        //Instigating the BankTransaction constructor, which has 500 records.
        BankTransaction[] mainArray = new BankTransaction[500];

        //This will come later on for the for loop, just declaring it at this time.

        //This will fill all the 500 records with random data, and print it out.
        for (int i = 0; i < 500; i++, BankTransaction.recordNumber++)
            {
                mainArray[i] = new BankTransaction(random.nextBoolean(), random.nextDouble() * (10000 - 0 + 1), random.nextInt(), random.nextLong());

                System.out.println("\n" + mainArray[i].getCanadianFunds() + "\n" + mainArray[i].getExchangeRate() + "\n" + mainArray[i].getTransactionNumber() + "\n" + mainArray[i].getTranscationReferenceNumber() +
                "\n" + (BankTransaction.recordNumber + 1) + "\n\n----------------------");
            }

        //Asking the user if he would like to run this section again.

        System.out.println("\nWould you like to run this code again? (true/false)\n");
        boolean answer = sc.nextBoolean();

        //This section will run the code again, filling the 500 records, overriding them.
        while (answer == true)
        {
            //Resetting the counter for the records.
            BankTransaction.recordNumber = 0;

            for (int i = 0; i < 500; i++, BankTransaction.recordNumber++)
            {
```

Part 2



```java
    {
        //This is the main method.
        Run | Debug
        public static void main(String[] args)
        {
            //Instigating the scanner and random.
            Scanner sc = new Scanner(System.in);
            Random random = new Random();

            //Instigating the BankTransaction constructor, which has 500 records.
            BankTransaction[] mainArray = new BankTransaction[500];

            //This will come later on for the for loop, just declaring it at this time.

            //This will fill all the 500 records with random data, and print it out.
            for (int i = 0; i < 500; i++, BankTransaction.recordNumber++)
                {
                    mainArray[i] = new BankTransaction(random.nextBoolean(), random.nextDouble() * (10000 - 0 + 1), random.nextInt(), random.nextLong());

                    System.out.println("\n" + mainArray[i].getCanadianFunds() + "\n" + mainArray[i].getExchangeRate() + "\n" + mainArray[i].getTransactionNumber() + "\n" + mainArray[i].getTranscationReferenceNumber() +
                    "\n" + (BankTransaction.recordNumber + 1) + "\n\n----------------------");
                }

            //Asking the user if he would like to run this section again.

            System.out.println("\nWould you like to run this code again? (true/false)\n");
            boolean answer = sc.nextBoolean();

            //This section will run the code again, filling the 500 records, overriding them.
            while (answer == true)
            {
                //Resetting the counter for the records.
                BankTransaction.recordNumber = 0;

                for (int i = 0; i < 500; i++, BankTransaction.recordNumber++)
                {
                    mainArray[i] = new BankTransaction(true, 999.999, 12345, 948346355);

                    System.out.println("\n" + mainArray[i].getCanadianFunds() + "\n" + mainArray[i].getExchangeRate() + "\n" + mainArray[i].getTransactionNumber() + "\n" + mainArray[i].getTranscationReferenceNumber() +
                    "\n" + (BankTransaction.recordNumber + 1) + "\n\n----------------------");
                }

                //Asking them once again, if they would like to keep running it. They can run it as many times as they wish.
                System.out.println("\nWould you like to run this code again? (true/false)\n");
                answer = sc.nextBoolean();
            }
        }
    }
```