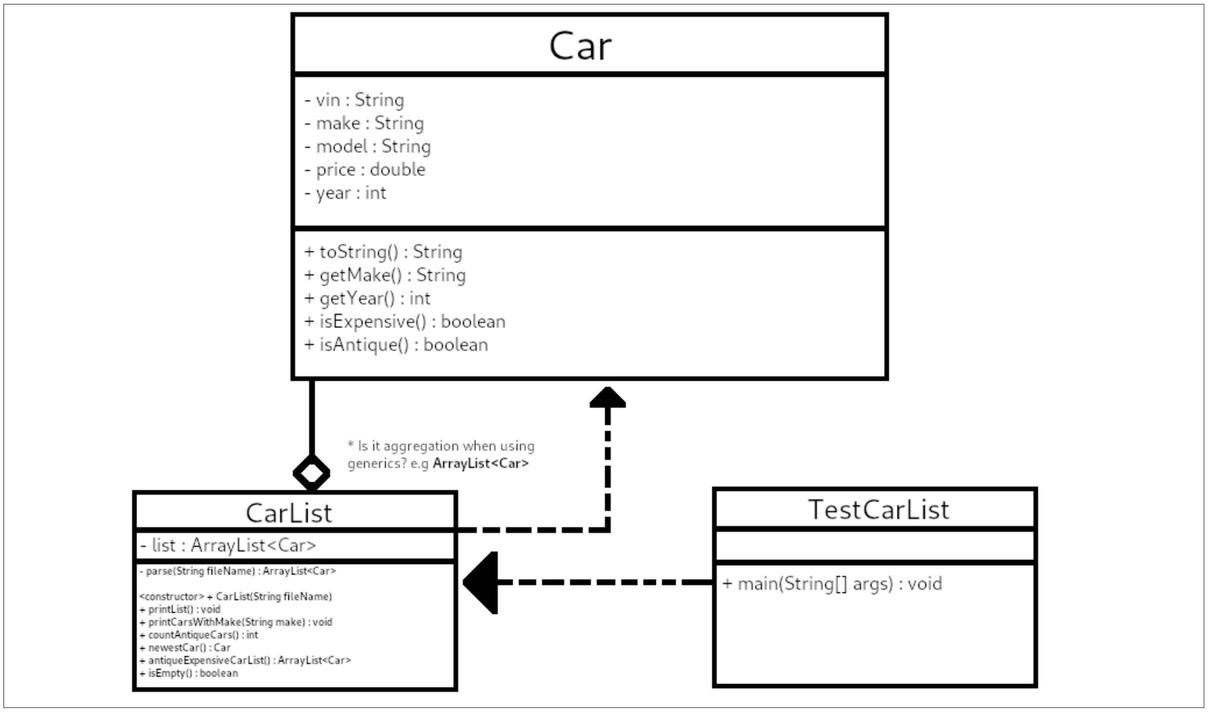
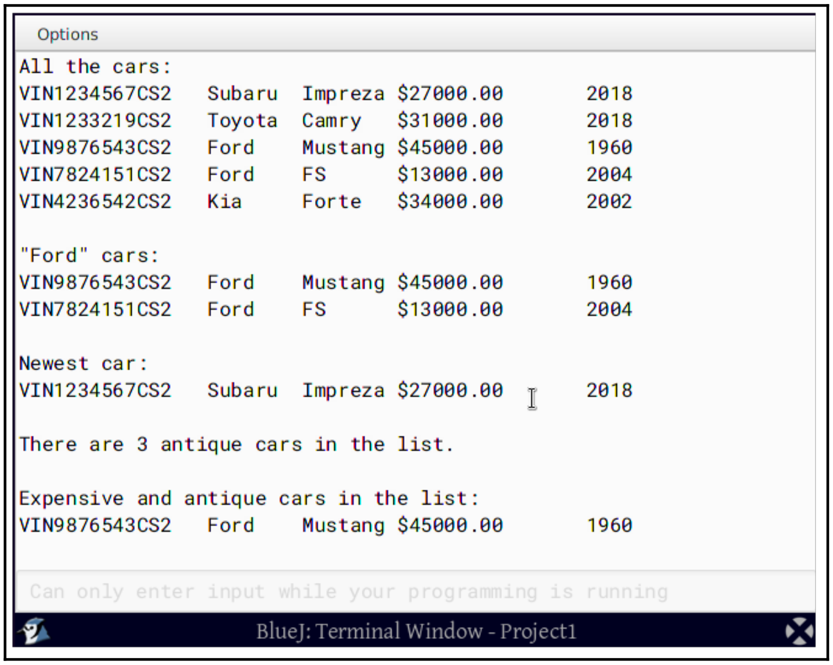
UML


Car
- vin : String
- make : String
- model : String
- price : double
- year : int
+ toString() : String
+ getMake() : String
+ getYear() : int
+ isExpensive() : boolean
+ isAntique() : boolean
* Is it aggregation when using generics? e.g **ArrayList<Car>**
CarList
- list : ArrayList<Car>
- parse(String fileName) : ArrayList<Car>
<constructor> + CarList(String fileName)
+ printList() : void
+ printCarsWithMake(String make) : void
+ countAntiqueCars() : int
+ newestCar() : Car
+ antiqueExpensiveCarList() : ArrayList<Car>
+ isEmpty() : boolean
TestCarList
+ main(String[] args) : void

Output


Options
Can only enter input while your programming is running
BlueJ: Terminal Window - Project1

```
All the cars:
VIN1234567CS2   Subaru  Impreza $27000.00       2018
VIN1233219CS2   Toyota  Camry   $31000.00       2018
VIN9876543CS2   Ford    Mustang $45000.00       1960
VIN7824151CS2   Ford    FS      $13000.00       2004
VIN4236542CS2   Kia     Forte   $34000.00       2002

"Ford" cars:
VIN9876543CS2   Ford    Mustang $45000.00       1960
VIN7824151CS2   Ford    FS      $13000.00       2004

Newest car:
VIN1234567CS2   Subaru  Impreza $27000.00       2018

There are 3 antique cars in the list.

Expensive and antique cars in the list:
VIN9876543CS2   Ford    Mustang $45000.00       1960
```

Car.java

```
/**
 * @author (Kyle Guarco)
 * @version (July 6, 2020)
 */
public class Car
{
    private String vin, make, model;
    private double price;
    private int year;

    public Car(String vin, String make, String model, double price, int year)
    {
        this.vin = vin;
        this.make = make;
        this.model = model;

        this.price = price;
        this.year = year;
    }

    @Override
    public String toString()
    {
        return String.format("VIN%s\t%s\t%s\t$%.2f\t%d",
                vin, make, model, price, year);
    }

    public String getMake()
    {
        return make;
    }

    public int getYear()
    {
        return year;
    }

    public boolean isExpensive()
    {
        return price > 30000;
    }

    public boolean isAntique()
    {
        return year < 1968;
    }
}
```

CarList.java

```
import java.util.ArrayList;
import java.util.Scanner;
import java.io.File;
import java.io.IOException;

/**
 * Parses and displays information from a list of cars.
 *
 * @author (Kyle Guarco)
 * @version (July 6, 2020)
 */
public class CarList
{
    private ArrayList<Car> list;

    public CarList(String listName)
    {
        this.list = parse(listName);
    }

    public void printList()
    {
        list.forEach(System.out::println);
    }

    public void printCarsWithMake(String make)
    {
        for (Car car : list)
            if (car.getMake().equalsIgnoreCase(make))
                System.out.println(car);
    }

    public int countAntiqueCars()
    {
        int count = 0;

        for (Car car : list)
            if (car.isExpensive())
                count++;

        return count;
    }

    public Car newestCar()
    {
        Car newest = list.get(0);

        for (Car car : list)
            if (car.getYear() > newest.getYear())
                newest = car;

        return newest;
    }

    public ArrayList<Car> antiqueExpensiveCarList()
    {
        ArrayList<Car> cars = new ArrayList<Car>();

        for (Car car : list)
            if (car.isAntique() && car.isExpensive())
                cars.add(car);

        return cars;
    }

    public boolean isEmpty()
    {
        return list.isEmpty();
    }

    private ArrayList<Car> parse(String fileName)
    {
        ArrayList<Car> cars = new ArrayList<Car>();

        try {
            Scanner scanner = new Scanner(new File(fileName));

            while (scanner.hasNextLine())
            {
                String line = scanner.nextLine();

                String[] args = line.split(" ");

                String  vin = args[0],
                        make = args[1],
                        model = args[2];
                int     year = Integer.parseInt(args[3]),
                        price = Integer.parseInt(args[4]);

                cars.add(new Car(vin, make, model, year, price));
            }
            // Don't forget to free up those valuable system resources!
            scanner.close();
        } catch (IOException e) {
            // Exit loudly: Tell the user the file doesn't exist and exit.
            System.out.println("The file containing the list of cars doesn't exist: " + fileName);
            System.exit(-1);
        }

        return cars;
    }
}
```

TestCarList.java

```java
/**
 *
 * @author (Kyle Guarco)
 * @version (July 6, 2020)
 */
public class TestCarList
{
    public static void main(String[] args)
    {
        // The fileName passed into CarList gets parsed using CarList.parse()
        CarList cars = new CarList("carList.txt");

        if (cars.isEmpty()) {
            // Do nothing if the file is empty.
            System.out.println("The file containing the cars is empty. Exiting...");
            System.exit(-2);
        }

        System.out.println("All the cars: ");
        cars.printList();

        System.out.println("\n\"Ford\" cars: ");
        cars.printCarsWithMake("Ford");

        System.out.println("\nNewest car: \n" + cars.newestCar());

        int antiqueCount = cars.countAntiqueCars();
        // If there are any antique cars in the list, print the count. Otherwise, print "no".
        Object anyAntique = antiqueCount > 0 ? antiqueCount : "no";
        System.out.println("\nThere are " + anyAntique + " antique cars in the list.");

        System.out.println("\nExpensive and antique cars in the list: ");
        cars.antiqueExpensiveCarList().forEach(System.out::println);
    }
}
```