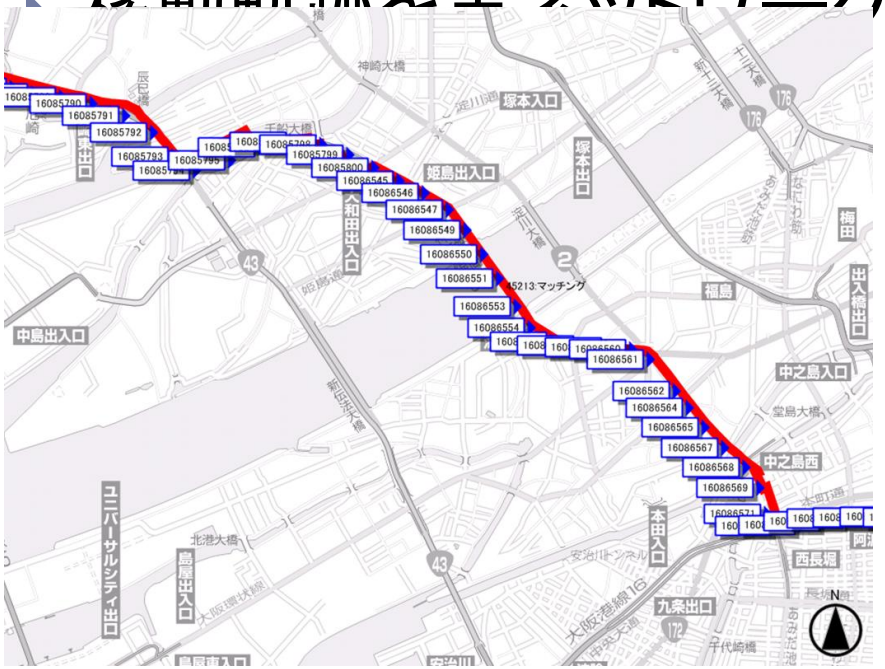
Eclipseによる
Javaプログラミングの基礎

単純な可視化（プロアトラス使用）

- 入力:トリップデータ，ロケーションデータ
- 出力:プロアトラスデータ
- 緯度経度データをトリップごとに地図形式に変換

マップマッチング

- 入力：trip，location，リンクデータ，ノードデータ
- 出力：地図データ，通過リンクデータ，通過経路データ
- 移動軌跡を実ネットワークに



	A	B	C	D	E	F	G	H	I
1	モニターID	ダイアリーI	交通手段	リンクID	リンク出発[	所要時間(s	速度(km/h	locationID	
2	he025	39344	100	306972	09:39.0	0	2.15E+08	0	
3	he025	39344	100	306967	09:39.0	9	34	14090443	
4	he025	39344	100	306956	09:48.0	36	9	14090446	
5	he025	39344	100	306944	10:24.0	10	39.6	14090447	
6	he025	39344	100	306932	10:34.0	10	25.2	14090448	
7	he025	39344	100	306933	10:44.0	11	24.8	14090450	
8	he025	39344	100	306624	10:55.0	57	9.2	14090453	
9	he025	39344	100	306625	11:52.0	16	38.4	14090454	
10	he025	39344	100	306184	12:08.0	10	57.6	14090455	
11	he025	39344	100	306185	12:18.0	10	15.8	14090456	
12	he025	39344	100	322094	12:28.0	0	2.15E+08	0	
13	he025	39344	100	322074	12:28.0	2	208.8	14090457	
14	he025	39344	100	319230	12:30.0	13	5.8	14090458	
15	he025	39344	100	319210	12:43.0	4	53.1	14090458	
16	he025	39344	100	319207	12:47.0	8	4.9	14090459	

	A	B	C	D	E	F	G	H	I
1	モニターID	ダイアリーI	経路出発n	経路到着n	所要時間(s	速度(km/h)			
2	he025	39344	100	3758100	139500	9730	1623	21.5	
3	he032	35635	100	1156200	2473400	16368	2822	20.8	
4	he032	36770	100	3428900	2474400	14802	3741	14.2	
5	he032	40098	100	2475700	1645100	16130	3204	18.1	
6	he045	42134	100	2164200	3632900	16031	1176	49	
7	he053	46183	100	15500	1199100	21610	1251	62.1	
8	he065	48696	100	2275200	15500	28395	1910	53.5	
9	he082	37513	100	1302300	2443900	32536	4291	27.2	
10	he094	35734	100	1459900	2560800	23926	3331	25.8	
11	he094	36498	100	1705700	2445600	24238	4630	18.8	
12	he094	36902	100	1459900	2560200	25320	4534	20.1	
13	he094	39306	100	1193500	2560800	24046	3831	22.5	
14	he094	39734	100	1193300	2161600	24263	3604	24.2	

PPデータ集計

- 入力：trip，location
- 出力：ツアーデータ，ツアー内トリップデータ

	A	B	C	D	E
1	ツアーID	モニターID	トリップ数		
2	29	sp20a	2		
3	85	sp02a	13		
4	98	sp02a	4		
5	102	sp02a	6		
6	111	sp02a	4		
7	115	sp02a	2		
8	117	sp02a	3		

	A	B	C	D	E	F	G	H	I	J	K	L
1	ツアーID	トリップID	モニターID	出発時刻	到着時刻	目的	代表交通手	出発地緯度	出発地経度	到着地緯度	到着地経度	
2	29	40720	sp20a	25:00.0	33:45.0	999	100	43.08791	141.2816	43.08799	141.2826	
3	29	40740	sp20a	44:54.0	59:30.0	200	100	43.0875	141.2832	43.08513	141.2587	
4	85	40040	sp02a	59:39.0	03:57.0	400	420	43.01684	141.45	43.01757	141.4533	
5	85	40041	sp02a	04:43.0	43:35.0	400	240	43.01765	141.4535	43.05475	141.3662	
6	85	40049	sp02a	45:32.0	14:38.0	400	411	43.05485	141.3654	43.05784	141.3523	
7	85	40053	sp02a	14:51.0	16:08.0	400	420	43.05736	141.3523	43.05767	141.3527	
8	85	40054	sp02a	19:04.0	20:11.0	999	420	43.05766	141.3529	43.05729	141.3527	
9	85	40056	sp02a	36:16.0	53:59.0	999	411	43.05889	141.3522	43.06571	141.3514	
10	85	40059	sp02a	58:05.0	02:52.0	999	420	43.06629	141.3511	43.06586	141.3491	
11	85	40060	sp02a	24:49.0	25:12.0	400	420	43.06784	141.3487	43.06795	141.3484	
12	85	40061	sp02a	34:38.0	53:26.0	200	420	43.06713	141.3508	43.06112	141.3471	

Eclipse

- 統合開発環境(Integrated Development Environment)
  - プログラミングのためのコンパイラ，テキストエディタ，デバッガなどを統合した開発環境.
- 主にJavaプログラミングに用いられる.
  - プラグインの導入で他の言語も編集可能
- リファクタリングやソースコード編集支援に優れている.
  - スペルミスの指摘や入力補助
- インストール
  - http://sourceforge.jp/projects/blancofw/wiki/blanco.eclipse.distribution.dev

からSetup.exeをダウンロードすると簡単.

Javaプロジェクトの作成

- プロジェクト
  - 一つのプログラムの基本単位
  - MapMatching, VirtualNetwork, DrmChangeなど
- プロジェクトの生成
  - [ファイル]→[新規作成]→[新規プロジェクト]→[Javaプロジェクト]
  - 外部のソースコードから作成する場合は[外部ソースからプロジェクトを作成]でプロジェクトフォルダの場所を指定する.
  - srcフォルダにソースコード(.java)が保存される.
  - ファイルのエンコードに注意（プロパティ→エンコード）
- 実行
  - Ctrl+F11でデバックなしで実行

プログラミングに関する注意事項

- { }ブロック内のインデントは必ず入れる.
  - Eclipseは勝手に挿入してくれる.
- 変数，インスタンス，関数名のつけ方
  - 最初は必ず小文字で始める. cf. クラス名は大文字で始める
  - 単語の区切りは大文字にする. もしくは_（アンダーバー）でつなぐ.
    - Ex) getData(), MAX_BUFFER_SPACE
- 演算子の前後はスペースを空ける.
  - y=a*x+b;→y = a * x + b;

オブジェクト指向

- “オブジェクト”と呼ばれる機能の部品でソフトウェアを構成させる.
- オブジェクト：ひとまとまりのデータ
  - フィールド: オブジェクトの状態を表す値
  - メソッド: オブジェクトの機能を表す.
  - 例）車オブジェクト
    - フィールド：全長，全幅，車高，排気量 etc…
    - メソッド：加速，減速，ギアチェンジ, etc…
- Javaではオブジェクトの型をクラスとして定義する.
- クラスの実体を表すのがインスタンス
  - 例）車クラスのインスタンスは「多摩　500　さ　12-34」の車

オブジェクト指向によるデータの格納

アクセス修飾子
メンバはprivate
メソッドはpublic

各メンバにはgetter, setter
メソッドでアクセスする.
※右クリック→[ソース]
→[getterおよびsetterの生成]

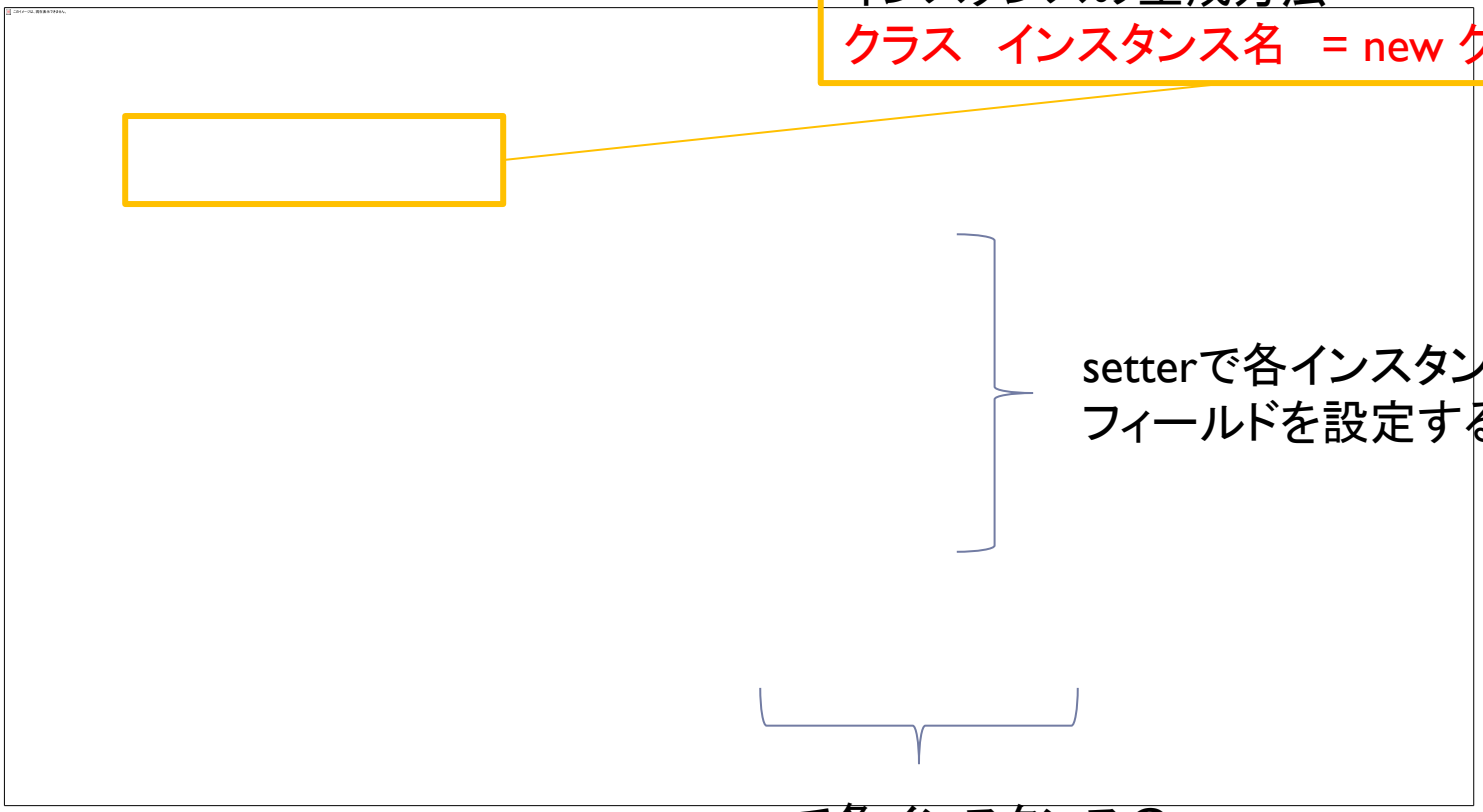
```
public class Car {
    private int length;
    private int width;
    private int hight;
    private int displacement;

    public int getLength() {
        return length;
    }
    public void setLength(int length) {
        this.length = length;
    }
    public int getWidth() {
        return width;
    }
    public void setWidth(int width) {
        this.width = width;
    }
    public int getHight() {
        return hight;
    }
    public void setHight(int hight) {
        this.hight = hight;
    }
    public int getDisplacement() {
        return displacement;
    }
    public void setDisplacement(int displacement) {
        this.displacement = displacement;
    }

}
```

フィールド

メソッド

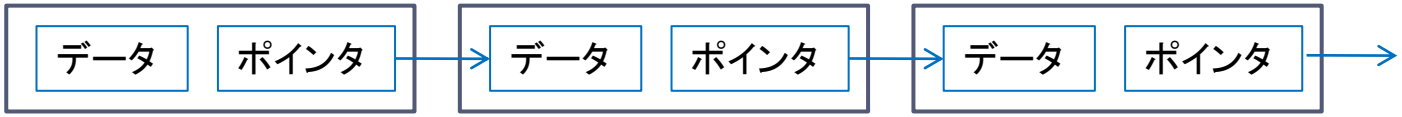
※1つのクラスは“*.java”という1つのファイルになる.

例）オブジェクト指向によるプログラミング例

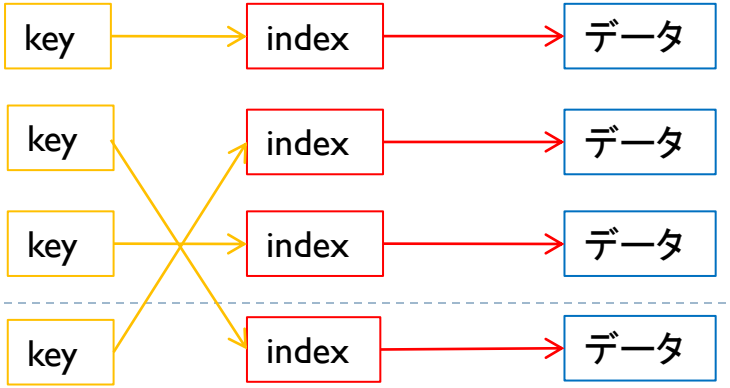


線形リストとハッシュテーブル

- データはインスタンスに格納される.
  - 膨大な数のインスタンスが生成される.
  - どうやって整理するか？
- 線形リスト：データと次のデータを指すポインタの配列



- ハッシュテーブル：キーをもとにしたハッシュ値が指すデータからキーとデータの対応づけをしたデータ構造.



ArrayList

- ArrayList：線形リストを実装したクラス.
  - List<クラス> インスタンス名 = new ArrayList<クラス>();
  - <クラス>に格納したいデータ型を指定する.

```
import java.util.*;//java.utilパッケージをインポートしておく

public class Test {
    public static void main(String[] args){
        List <Car> carList = new ArrayList<Car>();//ArrayListのインスタンスを生成

        for(int i = 0 ; i < 10 ; i++){//Carクラスのインスタンスを10個生成
            Car car = new Car();
            car.setNumber(i);//ナンバーを設定
            carList.add(car);//ArrayListに追加する.
        }

        for(int i = 0 ; i < carList.size() ; i++ ){
            Car car = carList.get(i);//ArrayListから順番に取り出す.
            System.out.println(car.getNumber());
        }

    }

}
```

addメソッドでデータを追加

getメソッドでデータを取り出す

HashMap

- HashMap: ハッシュテーブルを実装したクラス
  - Map<キー, クラス> インスタンス名
    = new HashMap <キー, クラス>();
  - キーにはStringやIntegerなどを指定することが多い.

```
import java.util.*;//java.utilパッケージをインポートしておく

public class Test {
    public static void main(String[] args){
        HashMap<Integer, Car> carMap = new HashMap<Integer, Car>();//HashMapのインスタンスを生成

        for(int i = 0 ; i < 10 ; i++){//Carクラスのインスタンスを10個生成
            Car car = new Car();
            car.setNumber(i);//ナンバーを設定
            carMap.put(i, car); //HashMapにキーとともに格納する.
        }

        for(int i = 0 ; i < carMap.size() ; i++ ){
            Car car = carMap.get(i);//HashMapにキーを指定して取り出す.
            System.out.println(car.getNumber());
        }

    }

}
```

putメソッドでキーと対応するデータを追加

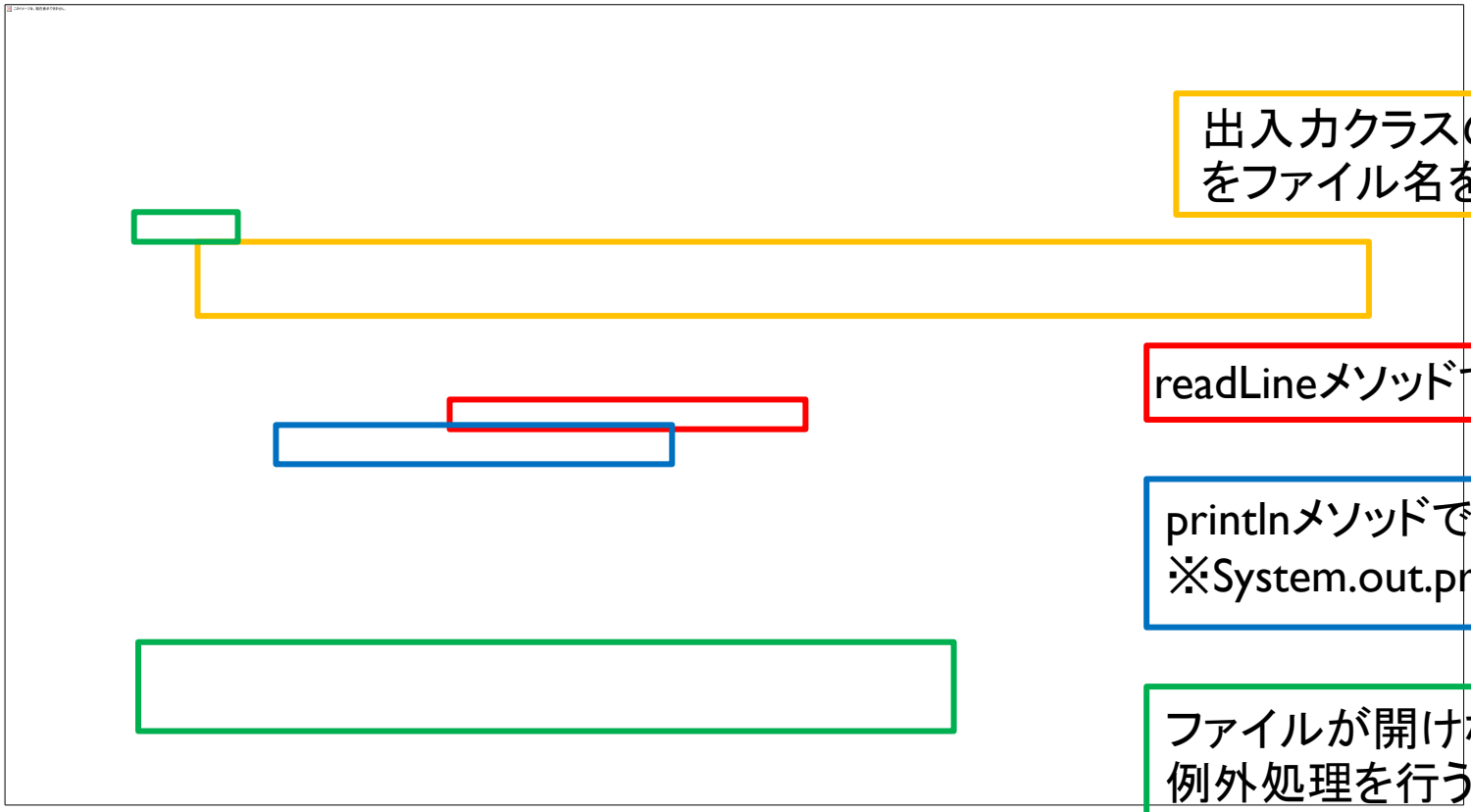
getメソッドでキーに対応するデータを取り出す

ファイル出入力

- プログラムの外部のデータを操作する.
  - 研究では，基本的にテキストファイル（エディタで開けるファイル）が操作できればよい.
  - データを1行ずつ（改行文字まで）読み書きするのが基本
- ファイル入力
  - BufferedReaderクラスを利用.
  - BufferedReader インスタンス
    = new BufferedReader(new FileReader(“ファイル名”));
- ファイル出力
  - PrintWriterクラスを利用
  - PrintWriter インスタンス
    = new PrintWriter(new FileWriter(“ファイル名”))

ファイル出入力例

- テキストデータをそのまま別のファイルにコピーする.



CSV形式のデータの読み込み

- CSV(Comma Separated Value): カンマ区切りのデータ
  - PPの生データは通常この形式
  - 例）

名前,性別,国語,数学
ジョン,男,87,50
マイケル,男,70,77

→1行のデータをさらに分解する必要がある.

- StringTokenizerクラスで分解する.
  - StringTokenizer インスタンス名
    = new StringTokenizer(分解対象文字列);
  - nextToken(*)で*に指定した区切り文字までの文字列を取り出す. 文字列の最後に到達すれば, NoSuchElementExceptionを投げる.

CSV形式のデータの読み込み

- CSVファイルをそのまま別のファイルにコピーする.

```
import java.io.*;//java.ioパッケージを読み込んでおく
import java.util.*;//java.utilパッケージを読み込んでおく

public class Test {
    public static void main(String[] args){
        String input = "test.csv";//String型でファイル名を指定する.
        String output = "copy.csv";

        try{
            BufferedReader inputLine = new BufferedReader(new FileReader(input));
            PrintWriter outputLine = new PrintWriter(new FileWriter(output));
            String line;

            //1行ずつ読み込み，String変数lineに格納する.
            while((line = inputLine.readLine() )!= null){
                StringTokenizer st = new StringTokenizer(line); //lineを分割の対象にする.
                String name = st.nextToken(",");//1回目の区切り（氏名）
                String sex = st.nextToken(",");//2回目の区切り（性別）
                String japanese = st.nextToken(",");//3回目の区切り（国語）
                String mathematics = st.nextToken(",");//4番目の区切り（数学）
                //PrintWriterのメソッドprintlnでファイルに出力
                outputLine.println(name+ "," + sex + "," + japanese + "," + mathematics);
            }

            //開いていた出入力ファイルを閉じる
            inputLine.close();
            outputLine.close();

        }catch(IOException e){
            System.out.println("ファイルが開けません");
        }
    }
}
```

名前,性別,国語,数学
ジョン,男,87,50
マイケル,男,70,77

分割する文字列を指定してインスタンス生成.

nextTokenメソッドを呼び出すごとに，区切り文字間の文字列を取り出す．引数に区切り文字を指定.

まとめのプログラム

- 以下の操作を行う.
  1. 車のデータが記入されたCSVファイルを読み込む.
  2. 読み込んだデータをCarクラスのインスタンスに格納する.
  3. Carクラスのインスタンスを線形リストにまとめる.
  4. 線形リストから順にインスタンスを取り出し，別のCSVファイルに書き込む.
- 読み込むデータ

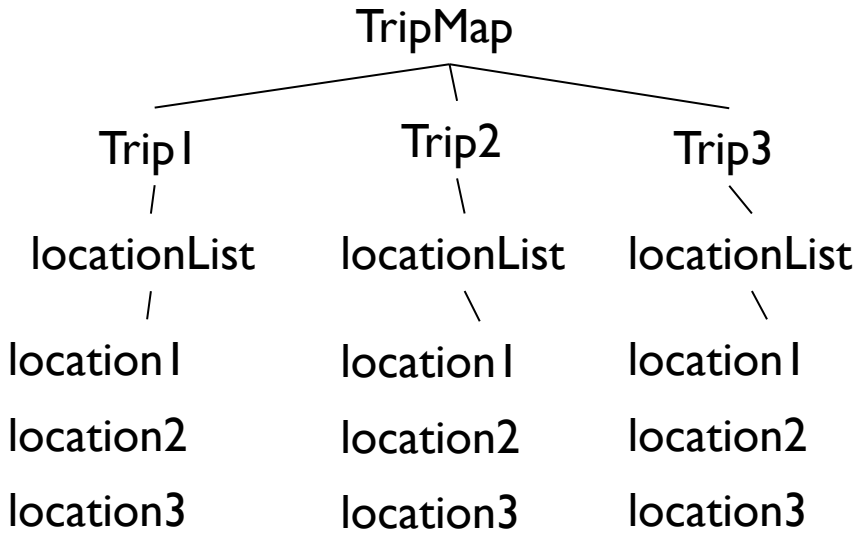
ナンバー	全長(mm)	全幅(mm)	車高(mm)	排気量(cc)
1122	5000	2500	1700	2000
2233	4500	2400	1600	1800
3344	4300	2400	1550	1750
4455	3000	2000	1500	1200
5566	4500	2500	1600	1500

```java
import java.io.*;//java.ioパッケージを読み込んでおく
import java.util.*;//java.utilパッケージを読み込んでおく

public class Test {
    public static void main(String[] args){
        List<Car> carList = new ArrayList<Car>();//データ格納用リスト
        String input = "carSample.csv";//String型でファイル名を指定する.
        String output = "copy.csv";

        try{
            BufferedReader inputLine = new BufferedReader(new FileReader(input));
            PrintWriter outputLine = new PrintWriter(new FileWriter(output));
            String line;

            line = inputLine.readLine();//ヘッダを読み込む
            outputLine.println(line);//ヘッダを書き込む

            //1行ずつ読み込み
            while((line = inputLine.readLine() )!= null){//String変数lineに格納する.
                StringTokenizer st = new StringTokenizer(line);//lineを分割の対象にする.

                Car car = new Car();//Carクラスのインスタンスを作成
                //フィールドの設定を行う.
                car.setNumber(Integer.parseInt( st.nextToken(",")));
                car.setLength(Integer.parseInt( st.nextToken(",")));
                car.setWidth(Integer.parseInt( st.nextToken(",")));
                car.setHight(Integer.parseInt( st.nextToken(",")));
                car.setDisplacement(Integer.parseInt( st.nextToken(",")));

                carList.add(car);//リストに追加する
            }

            //carListの中身を順に出力する.
            for(Car car : carList){
                outputLine.print(car.getNumber() + ",");
                outputLine.print(car.getLength() + ",");
                outputLine.print(car.getWidth() + ",");
                outputLine.print(car.getHight() + ",");
                outputLine.println(car.getDisplacement());
            }

            //開いていた出入力ファイルを閉じる
            inputLine.close();
            outputLine.close();

        }catch(IOException e){
            System.out.println("ファイルが開けません");
        }
    }
}
```

Integer.parseIntでString型をInteger型に変換する.

PlotProatrasの構造

- ファイルの読み込み
- （同時に）トリップ，ロケーションデータの格納
- ファイルのプロアトラス形式での書き出し
- だけ！



- 格納イメージ
- クラスは属性を規定する
- Tripクラス
  - トリップID，ロケーションリスト
- Locationクラス
  - LocationID，トリップID，ユーザーID，緯度，経度など

PlotProatrasの構造

- 使用するクラス
  - 自作
  - Location　:ロケーションデータ属性を設定
  - Trip　:トリップデータ属性を設定
  - Henkan　:データの形式を変換する（日本測地→世界測地など）
  - Output　:データを出力する

  - FileReader　:データ読み込み
  - BuffardReader
  - PrintWriter:データ書き出し
  - FileWriter
  - Integer　:整数を扱う
  - Double　:小数を扱う
  - Timestamp　:時間を扱う

課題：ソースコードリーディング

- PPデータを読み込んで，形式を変換して書き出しを行うプログラム
  1. CSVファイルの読み込み．
  2. PPデータの格納（Tripクラス．ArrayList利用）
  3. CSVファイルの書き出し．（トリップID,目的のみ）
- 課題
  1. ソースコードを読み，プログラムの内容を理解する．
  2. ソースコードにコメントをつける．穴埋め形式．
  3. 出力データを増やし，Excelでクロス集計を行う．

※今回取り扱った範囲外のテクニックも含まれる．

Tips

- デバッグ
  1. 問題が発生していると思われる個所にブレークポイントを置く．場合によってはブレークポイントを右クリックし，[プロパティ]→[条件]から条件を設定．
  2. デバッグを押し，ブレークポイントに止まるまでプログラムを実行する．変数の中身は[インスペクション]で確認できる．
- Eclipseのメモリ解放
  - ウィンドウ⇒設定⇒インストール済みJRE⇒編集⇒デフォルトVM引数を「-Xmx・・・M」に設定．・・・は任意に数字を指定．
- 要素コメントの付け方
  - クラス・関数名にカーソルを合わせ，右クリック→[ソース]→[要素コメントの作成]