Some suggestions about patient class

We can set the instance variables of patient class as below

```
public class Patient {

  //variables from PDF and table 1.
  Integer patientNbr;
  String race;
  Integer gender; //Female = 1 and Male = 0
  Integer ageCat; //Age category 1 to 10
  Integer encounterTotal;
  double avgNumInpVisits;
  double avgNumProcedures;
  double avgNumLabProcedures;
  double avgNumMedications;
  double avgNumOutpatientVisits;
  double avgNumEmergencyVisits;
  Integer totA1CElevated = 0;
  Integer totReadmissions = 0;

}
```

Abovementioend changes will help me with 'similarity' calculation. I cannot use `string` datatype for the 'distance' calculation. The `string` variables needs to be dummy coded for calculation. We can handle this right in the place class.

Also, I would suggest to add follwoing `getter` method

```
 public Double[] getProfile(){
        Double[] profile = new Double[]{(double) this.getAge_category(),
                this.getAvg_num_lab_procedures(), this.getAvg_num_procedures(),
                this.getAvg_num_outpatient(), this.getAvg_num_inpatient(),
                this.getAvg_num_emergency(), (double) this.getTot_a1c_elevated(),
                this.getAvg_num_meds(), (double) this.getGender()}
    return profile;
     }
```

Again, this will help me get the necessary information for the calcualtions from instance of `Patient` class.

Separate `processor` class

We can only keep the instance variables and `getter` methods in the `Patient` class.

New `patientProcessor` class, can create instance of `HashMap<Interger,Patient>` and iterate through `Arraylist<ClinicalEncounter>`.

This can be part of top level method of `patientProcessor` class

```
public HashMap<Integer,Patient> buildPatientProfiles(ClinicalEncounter clinic){

    HashMap<Integer, Patient> patientsMap = new HashMap<Integer, Patient>();

    //Loop through encounters list and update or add patient data.
    for(ClinicalEncounter encounter : clinic) {
      if(patientsMap.containsKey(encounter.getPatientNbr())) {
        this.patientUpdate(encounter);
      }

      //Call overloaded constructor to create new patient
      else {
        Patient workingPatient = new Patient(encounter);
        //As we dont have constructor anymore for Patient Class, you can build it
here
    workingPatient.updateAge(encounter);
    workingPatient.patientNbr = encounter.getPatientNbr();
    workingPatient.race = encounter.getRace();
    workingPatient.gender = encounter.getGender();
    workingPatient.encounterTotal = 1;
    workingPatient.avgNumInpVisits = encounter.getNumberInpatient();
    workingPatient.avgNumProcedures = encounter.getNumProcedures();
    workingPatient.avgNumLabProcedures = encounter.getNumLabProcedures();
    workingPatient.avgNumMedications = encounter.getNumMedications();
    workingPatient.avgNumOutpatientVisits = encounter.getNumberOutpatient();
    workingPatient.avgNumEmergencyVisits = encounter.getNumberEmergency();
    workingPatient.calcA1CResults(encounter);
    workingPatient.reAdmit(encounter)
      }
    }
  }
```

Then you can keep most of your existing methods from `Patient` class and carryover to new `processor` class.