Methodology session

Constraint tracing procedure (cheat sheet)

1. Read and understand constraint
2. Find **enforcing statement(s)**
3. Identify data used in enforcing statement(s)
4. Find the definition of data used in enforcing statement(s)

DOs and DON’Ts

- DO select all lines if statements are split
- DO reach out if constraint does not seem to be implemented
- DO make sure that the constraint check/enforcement actually corresponds to the scenario
- DO follow LoC trace format in spreadsheet
- DON’T mark places where method with enforcing statement is being called (unless they define data used in enforcing statement)
- DON’T consider test files in constraint implementation
- DON’T consider library code in constraint implementation (only calls to it if applicable, see Data definition: method call)

Example constraint

We will use the following example constraint and go through the steps:

> *Patient age must be greater than 18 years old*

1. Read and understand the constraint

- What are the entities involved?
- Patient age must be greater than 18 years old
  - Entity: Patient
  - Attribute: Patient.age
  - Other: constant: 18

2. Find enforcing statement(s)

- You must start by finding the statement where the checking or enforcement happens (the **enforcing statement(s)**)
- Enforcing statement is always part of the trace
- If checking/enforcement is delegated to library/framework, enforcing statement is method call where this happens, e.g.:
  - library.setDefaultForField(“firstName”, “John Doe”)
  - library.checkValid(person.getName())

Example of enforcing statement

- Constraint: *Patient age must be greater than 18 years old*

```
if(patient.getAge() > 18)
```

```
return patient.getAge() > 18;
```

```
boolean result = patient.getAge() > 18;
```

```
if(age > 18)
```

```
if(patientAge > EIGHTEEN)
```

Could be opposite condition as well → `if(patientAge <= 18)`

Example of enforcing statement

- Enforcing statement is always part of the trace

```
if(patient.getAge() > 18)
```

```
return patient.getAge() > 18;
```

```
boolean result = patient.getAge() > 18;
```

```
if(age > 18)
```

```
if(patientAge > EIGHTEEN)
```

```
if(patientAge <= 18)
```

3. Identify data used in enforcing statement

- What are the variables and constants used in the check?
- But **only** those related to current constraint

```
if(patient.getAge() > 18)
```

```
if(age > 18)
```

```
if(patientAge > EIGHTEEN)
```

```
if(patient.getAge() > 18 || patient.hasMoney())
```

hasMoney is not part of this constraint

4. Find the definition of data used in enforcing statement

- For **each** variable and constant identified in enforcing statement
- Definition can be one of:
  1. Inline
  2. Field definition
  3. Method definition
  4. Method call
  5. Variable definition
- Make sure that you select the right one for each case!
- The definition for each variable and constant will be part of the trace along with enforcing statement

Data definition: Inline

- ONLY when a constant is defined in the same statement where it is used
- In this case, no additional lines need to be selected for this constant, only enforcing statement
- If not defined on enforcing statement, statement defining it must be selected

```
if(patient.getAge() > 18)
```

```
if(name.equals("something"))
```

Data definition: Field definition

- ONLY when a field value is being used **unchanged**

```
if(patient.getAge() > 18)
```



```
private int age;

public int getAge() {
    return age;
}
```

```
if(patient.age > 18)
```



```
public int age;
```

```
public static final int EIGHTEEN = 18;
...

if(patient.getAge() > EIGHTEEN)
```

A constant field being used unchanged fits this case as well

Data definition: Method definition

- ONLY when a field value is being **calculated** in a method, or has **multiple definitions** inside that method
- In this case, method **definition** is part of the trace

Calculated in method

```
if(patient.calculateAge() > 18)
```



```
private int birthYear;

public int calculateAge() {
    return currentYear- birthYear;
}
```

Multiple definitions

```
if(patient.getAge() > 18)
```



```
public int getAge() {
    if(age != null) {
        return age;
    } else {
        return currentYear - birthYear ;
    }
}
```

Data definition: Method call

- ONLY when the data is accessed from **outside application code**
- In this case, method **call** is part of the trace

If database.getAge is external (framework or library)

```
if(patient.getAge() > 18)
```



```
public int getAge() {
    return database.getAge();
}
```

If request.getValue is framework or library method

```
int age = request.getValue("age")

if(age > 18)
```

Data definition: Variable definition

- ONLY when a literal is being defined inside the method body
- In this case, variable definition is part of the trace

```
int eighteen = 18

if(age > eighteen)
```

4. Find the definition of data used in enforcing statement

- Definition can be one of:
  1. Inline
  2. Field definition
  3. Method definition
  4. Method call
  5. Variable definition
- If the definition does not seem to be any of these, reach out to me

Revised constraint tracing procedure (cheat sheet)

- Always follow this procedure when tracing a constraint
  1. Read and understand constraint
  2. Find enforcing statement(s)
  3. Identify data used in enforcing statement(s)
  4. Find the definition of data used in enforcing statement(s)

Example 1

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

```
public static void main(String[] args) {
    ...
    int patientAge = patient.getAge();
    int limit = 18;

    if (patientAge > limit) {
        ...
    }
}
```

Example 1

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

```
public static void main(String[] args) {
    ...
    int patientAge = patient.getAge();
    int limit = 18;

    if (patientAge > limit) {
        ...
    }
}
```

Enforcing statement

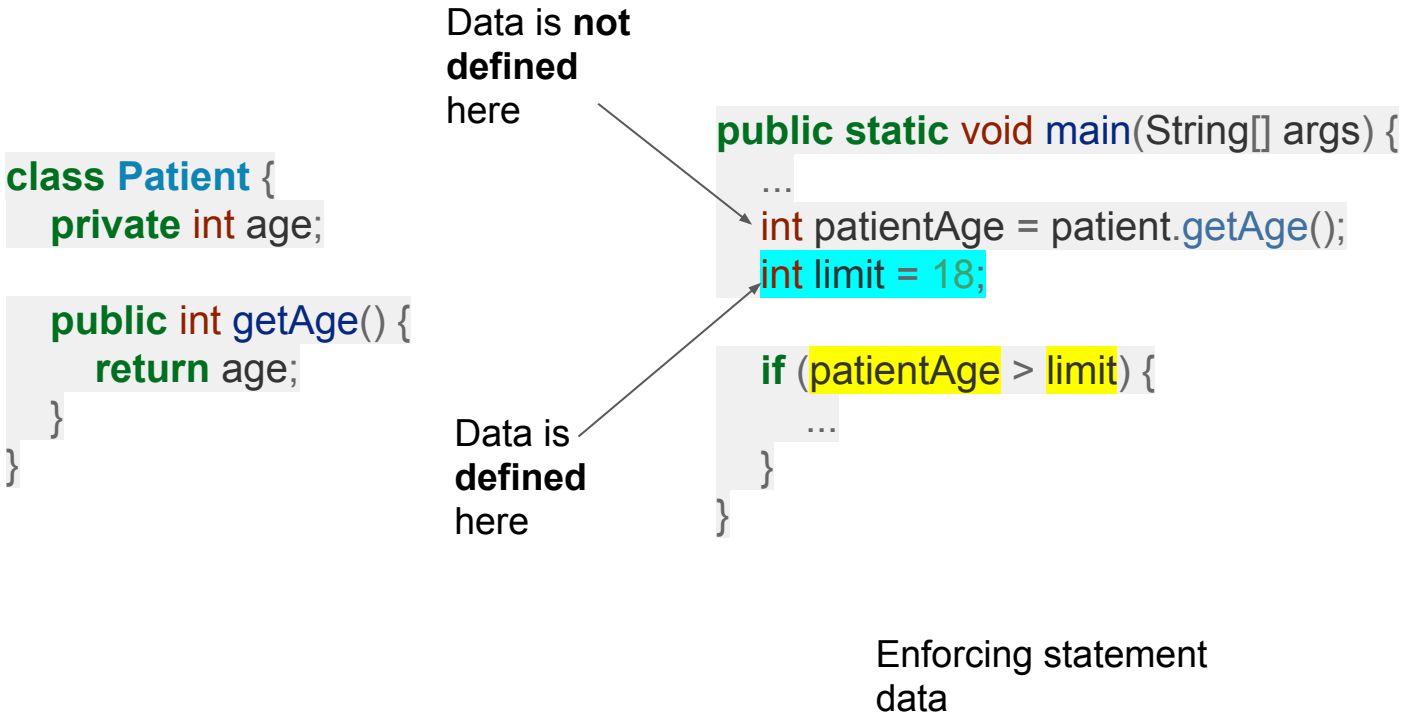
Example 1

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

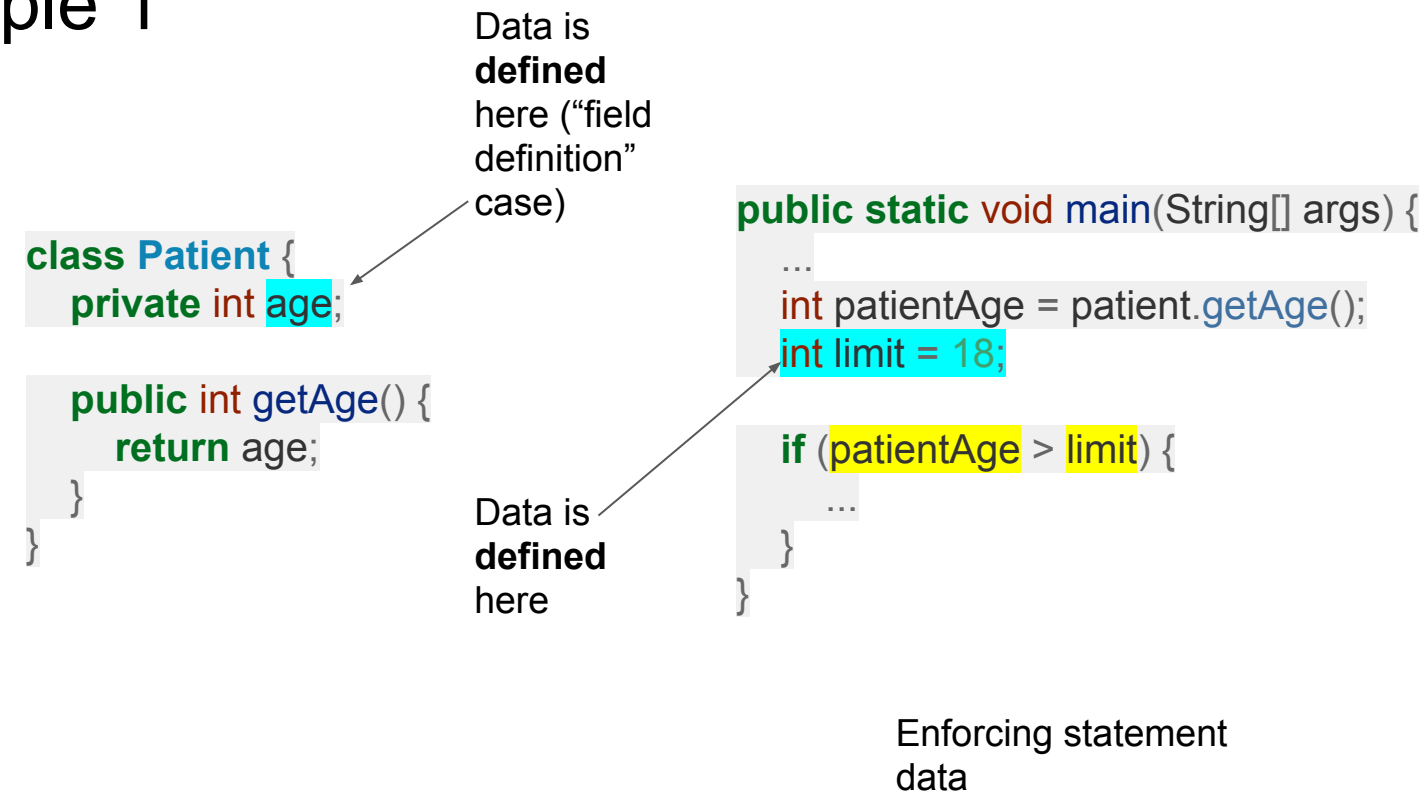
```
public static void main(String[] args) {
    ...
    int patientAge = patient.getAge();
    int limit = 18;

    if (patientAge > limit) {
        ...
    }
}
```

Enforcing statement data

Example 1


Data is **not defined** here
class Patient {
private int age;
public int getAge() {
return age;
}
}
public static void main(String[] args) {
...
int patientAge = patient.getAge();
int limit = 18;
if (patientAge > limit) {
...
}
}
Data is **defined** here
Enforcing statement data

Example 1


Data is **defined** here (“field definition” case)
class Patient {
private int age;
public int getAge() {
return age;
}
}
public static void main(String[] args) {
...
int patientAge = patient.getAge();
int limit = 18;
if (patientAge > limit) {
...
}
}
Data is **defined** here
Enforcing statement data

Example 1

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

```
public static void main(String[] args) {
    ...
    int patientAge = patient.getAge();
    int limit = 18;

    if (patientAge > limit) {
        ...
    }
}
```

Final trace

Example 2

```
public String[] OPTIONS = new String[] {...};

public static void main(String[] args) {
    ...

    String selection = httpLibrary.getSelection();

    for (String option: OPTIONS) {
        if (selection.equals(option)) {
            return;
        }
    }

    throw new Exception("Invalid option");
}
```

Example 2

```
public String[] OPTIONS = new String[] {...};

public static void main(String[] args) {
    ...

    String selection = httpLibrary.getSelection();

    for (String option: OPTIONS) {
        if (selection.equals(option)) {
            return;
        }
    }

    throw new Exception("Invalid option");
}
```

Example 3

```
public static void main(String[] args) {
    ...

    boolean result = checkAge(patient, 18);

    if(result) {
        ...
    }
}
```

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

```
public boolean checkAge(Patient patient, int ageLimit) {
    return patient.getAge() > ageLimit;
}
```

Example 3

```
public static void main(String[] args) {
    ...

    boolean result = checkAge(patient, 18);

    if(result) {
        ...
    }
}
```

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

```
public boolean checkAge(Patient patient, int ageLimit) {
    return patient.getAge() > ageLimit;
}
```

Example 3

```
public static void main(String[] args) {
    ...

    boolean result = checkAge(patient, 18);

    if(result) {
        ...
    }
}
```

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

```
public boolean checkAge(Patient patient, int ageLimit) {
    return patient.getAge() > ageLimit;
}
```

Example 3

```
public static void main(String[] args) {
    ...

    boolean result = checkAge(patient, 18);

    if(result) {
        ...
    }
}
```

```
class Patient {
    private int age;

    public int getAge() {
        return age;
    }
}
```

```
public boolean checkAge(Patient patient, int ageLimit) {
    return patient.getAge() > ageLimit;
}
```