

Mid-Term Exam - Specification 1 of 5

Create a report that writes records for all medical claims that are **in-policy**.
Use the record layout shown here →

Glossary of Terms:

- **Policy-Number:** Unique identifying number for each policy sold
- **Policy-Type:** Type of insurance in-force
- **Policy-Benefit-Date:** The date the policy was written
- **Policy-Amount:** The amount currently left on the policy
- **Policy-Deductible:** The amount paid for covered services before insurance kicks in
- **Policy-Coinsurance:** The % costs of a covered health service paid by the insured
- **Claim-Amount:** The amount of the medical charges
- **Claim-Amount-Paid:** The amount the company will cover of a claim
- **Max-Coverage-Amount:** The max claim amount for a policy of this type: \$999,999.99

| | | |
|---------------------|----------------------------|------------------------------------|
| COBOL Record Layout | 01 CLAIM-RECORD-WS. | |
| | 05 INSURED-DETAILS. | |
| | 10 INSURED-POLICY-NO | PIC 9(07). |
| | 10 INSURED-LAST-NAME | PIC X(15). |
| | 10 INSURED-FIRST-NAME | PIC X(10). |
| | 05 POLICY-DETAILS. | |
| | 10 POLICY-TYPE | PIC 9. |
| | 88 PRIVATE | VALUE 1. |
| | 88 MEDICARE | VALUE 2. |
| | 88 AFFORDABLE-CARE | VALUE 3. |
| | 10 POLICY-BENEFIT-DATE-NUM | PIC 9(08). |
| | 10 POLICY-BENEFIT-DATE-X | REDEFINES |
| | | POLICY-BENEFIT-DATE-NUM PIC X(08). |
| | 10 POLICY-BENEFIT-PERIOD | REDEFINES |
| | | POLICY-BENEFIT-DATE-NUM. |
| | 15 POLICY-YEAR | PIC 9(04). |
| | 15 POLICY-MONTH | PIC 9(02). |
| | 15 POLICY-DAY | PIC 9(02). |
| | 10 POLICY-AMOUNT | PIC S9(7)V99. |
| | 10 POLICY-DEDUCTIBLE-PAID | PIC S9(4). |
| | 10 POLICY-COINSURANCE | PIC V99. |
| | 05 CLAIM-DETAILS. | |
| | 10 CLAIM-AMOUNT | PIC S9(7)V99. |
| | 10 CLAIM-AMOUNT-PAID | PIC S9(7)V99. |
| | 05 FILLER | PIC X(6). |


| | | | | | | | |
|----------|----------|--------|-----------|------------|-------|----------------|------------|
| 11111111 | Gribou | Andre | 120181224 | 1111111111 | 11111 | 11111111111111 | 1111111111 |
| 22222222 | McKernan | Joseph | 220191225 | 2222222222 | 22222 | 22222222222222 | 2222222222 |
| 22222222 | Mariano | Thomas | 320201226 | 3333333333 | 33333 | 33333333333333 | 3333333333 |
| | 1 | | 2 | | 3 | | 4 |
| | 5 | | 6 | | 7 | | 8 |

Sample
Data

IF CLAIM-AMOUNT < ALLOWED-AMT

Calculate Deductible:

The Deductible is the Policy-Amount times the company's deductible (copay) % which is a constant **.02**

 **Policy-Deductible-Paid** is the amount the Insured has already paid towards their deductible and is carried in the data.

Process the claim:

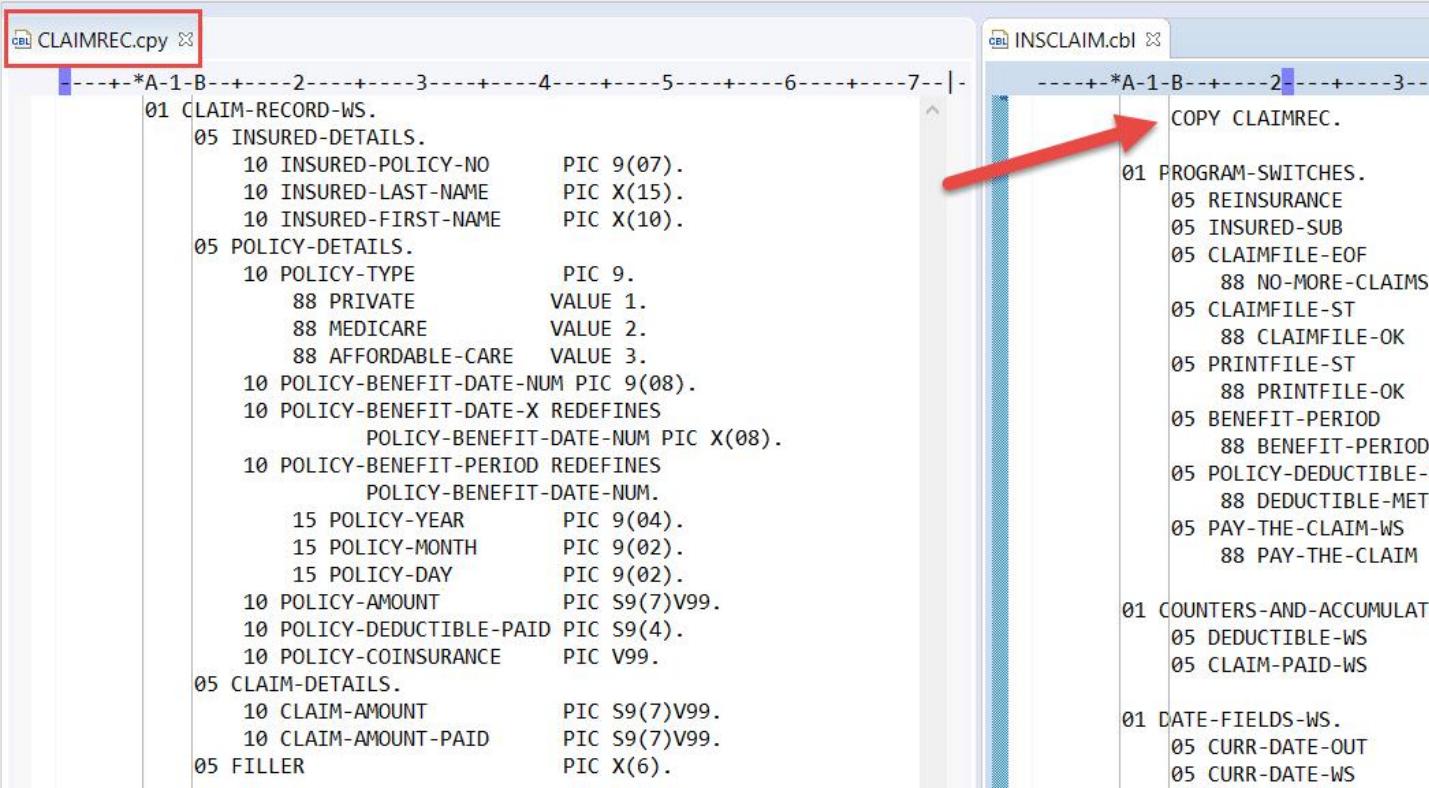
- If the Deductible has been met the Claim-Paid will be:
 $\text{Claim-Amount} \text{ minus the } (\text{Policy-Coinsurance} * \text{Claim-Amount})$
- If the Deductible has not been met the Claim-Paid will be the
 $\text{Claim-Amount} - \text{Deductible} - (\text{Policy-Coinsurance} * \text{Claim-Amount})$

- Finally, subtract the Claim-Paid from the Policy-Amount (which is whatever is left for this insured). If the result is > zero pay the claim (print the record) otherwise deny the claim (don't print the record on the report)

Mid-Term Exam Spec: 3 of 5 - COBOL Copybooks

A copybook is a file of reusable source code. In most cases the Copybook will be a data record that maps exactly to the fields in a file (i.e. the record layout). Copybooks are stored in Copy Libraries. Most production mainframe applications make heavy use of Copybooks.

Here's your first one...Note that you should see this in your <TSOID>.COBOL.COPYLIB



```
CLAIMREC.cpy
-----*A-1-B-----2-----3-----4-----5-----6-----7--|
01 CLAIM-RECORD-WS.
  05 INSURED-DETAILS.
    10 INSURED-POLICY-NO      PIC 9(07).
    10 INSURED-LAST-NAME     PIC X(15).
    10 INSURED-FIRST-NAME    PIC X(10).
  05 POLICY-DETAILS.
    10 POLICY-TYPE            PIC 9.
      88 PRIVATE              VALUE 1.
      88 MEDICARE             VALUE 2.
      88 AFFORDABLE-CARE      VALUE 3.
    10 POLICY-BENEFIT-DATE-NUM PIC 9(08).
    10 POLICY-BENEFIT-DATE-X REDEFINES
      POLICY-BENEFIT-DATE-NUM PIC X(08).
    10 POLICY-BENEFIT-PERIOD REDEFINES
      POLICY-BENEFIT-DATE-NUM.
      15 POLICY-YEAR          PIC 9(04).
      15 POLICY-MONTH         PIC 9(02).
      15 POLICY-DAY           PIC 9(02).
    10 POLICY-AMOUNT          PIC S9(7)V99.
    10 POLICY-DEDUCTIBLE-PAID PIC S9(4).
    10 POLICY-COINSURANCE     PIC V99.
  05 CLAIM-DETAILS.
    10 CLAIM-AMOUNT          PIC S9(7)V99.
    10 CLAIM-AMOUNT-PAID     PIC S9(7)V99.
  05 FILLER                  PIC X(6).

INSCCLAIM.cbi
-----*A-1-B-----2-----3-----
COPY CLAIMREC.
01 PROGRAM-SWITCHES.
  05 REINSURANCE
  05 INSURED-SUB
  05 CLAIMFILE-EOF
    88 NO-MORE-CLAIMS
  05 CLAIMFILE-ST
    88 CLAIMFILE-OK
  05 PRINTFILE-ST
    88 PRINTFILE-OK
  05 BENEFIT-PERIOD
    88 BENEFIT-PERIOD
  05 POLICY-DEDUCTIBLE-I
    88 DEDUCTIBLE-MET
  05 PAY-THE-CLAIM-WS
    88 PAY-THE-CLAIM
01 COUNTERS-AND-ACCUMULAT
  05 DEDUCTIBLE-WS
  05 CLAIM-PAID-WS
01 DATE-FIELDS-WS.
  05 CURR-DATE-OUT
  05 CURR-DATE-WS
```

The COBOL statement used to access a Copybook - from which the program can reference the fields in the record layout it contains - you code:

COPY <copybookname> .
.....in the B margin

Mid-Term Exam Spec: 4 of 5 - Output Report

- Below is a sample output from the report produced by an example-program from a student in a previous class.
- You are to recreate this output using the:
 - Record Layout
 - Insurance Claims Logic
 - Copybook
- Note that you're encouraged to make up your own data (MidTerm Slide 1 of 5) - so that you lock down the FileData ↔ Copybook layout field & byte positions

| | | | | | | | | | |
|--|------------------|---------------|--------------|---------------|--------------|------------------|-----------------|-----------------|---------------|
| CBL INSCCLAIM.cbl DDS0001.RUNCOB12.JOB06130.D0000108.?.spool | | | | | | | | | |
| -----1-----2-----3-----4-----5-----6-----7-- --8 | | | | | | | | | |
| 2020/05/24 Group Claims Daily Totals | | | | | | | | | |
| POLICY TYPE | POLICY NUMBER | FIRST NAME | LAST NAME | RENEW DATE | DEDUC MET | COPAY PERCENT | DEDUC AMOUNT | CLAIM AMOUNT | CLAIM PAID |
| EMPLOYER-PRIVATE | 1-111-11 | PAUL | NEWTON | 2018/12/24 | N | .002 | \$222 | \$1,111,111.11 | \$1,111.11 |
| STANDARD MEDICARE | 2-222-22 | MARTIN | KEENAN | 2019/12/25 | N | .002 | \$444 | \$2,222,222.22 | \$2,222.22 |
| AFFORDABLE CARE ACT | 2-222-22 | BILLY | MARTIN | 2020/12/26 | N | .002 | \$666 | \$3,333,333.33 | \$3,333.33 |

Mid-Term Exam Spec: 5 of 5 - Attacking this kind of programming challenge

Level 1 Development - Understand the problem and design the solution "from a 1,000 foot view"

- Create the test data - first. By doing this you'll familiarize yourself with the program's "problem space"
- Study the claims processing business functionality When you understand exactly "what" is required, break the whole into parts ... down to what COBOL paragraphs will be required - and what will be in the paragraphs.
- Create a new program from an existing program - one that reads a file and writes a report so that you can inherit the ENVIRONMENT DIVISION and other COBOL boilerplate. Substitute the USERID.LEARN.INSCLAIM file (ENVIRONMENT DIVISION) for the existing input file and use the CLAIMREC Copybook file. Liberal use of Find/Replace All.
- Code the PROCEDURE DIVISION logic to PERFORM each paragraph. If using ZOD/IDz use the Program Control Flow to visualize your work.
- Comment each of the paragraphs with a "Flower Box" above the para-name, describing the logic in the paragraph
- STUB OUT the lower-level COBOL paragraphs that calc the deductible and process claims
 - STUB OUT means to code the paragraph names, but don't add anything except a DISPLAY statement: **DISPLAY 'PARA ' <paragraph-name>.**

Level 1 Testing - COBUCLG - or COBUCLD the program, and view the logic flow for accuracy

Level 2 Development: Divide and conquer - Create the rest of the rest of the program. Code the PROCEDURE DIVISION logic to open the file. And read and write each record.

- For MOVE & COMPUTE statements, copy the input record (CLAIMREC) into your processing routine - just for the sake of editing speed. Consistent naming conventions are crucial throughout the code.
 - Name Working-Storage fields for the same domain the same as the input/output fields - but use a WS- prefix or suffix

Level 2 Testing - Test/Verify then move on → one primary function at a time.