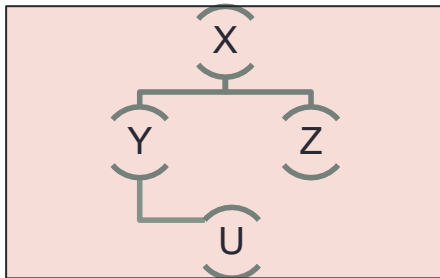


# Seminar 04. Continut

- **Niveluri de testare. Exemplu**
  - Testare unitară
  - Testare de integrare
  - Tipuri de bug-uri
- **Problemă**

# Niveluri de testare. Exemplu

- Se consideră următoarea diagrama de dependență:



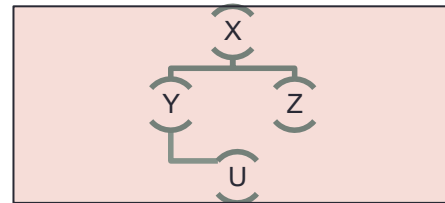
- Se cere să se simuleze testarea de integrare pentru modulele X, Y, Z și U folosind diferite strategii:
  - non-incrementală: **big-bang**;
  - incrementală: **top-down (depth first, breath first), bottom-up**;
  - mixtă: **sandwich**.

# Niveluri de testare. Module utilizate în testare

- La testarea unitară și testarea de integrare se consideră următoarele tipuri de module necesare pentru a testa X, Y, Z și U:

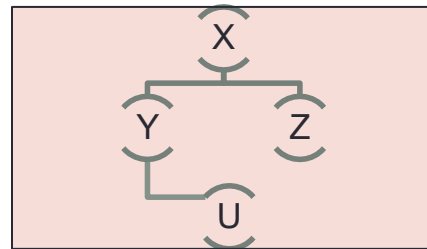
- driver** pentru modulele X, Y și Z –  $D_X, D_Y, D_Z, D_U$ ;
- stub** (mock, fake, dummy, spy) pentru modulele Y și Z –  $S_Y, S_Z, S_U$ .

- vezi [Lab04\\_TutorialUnitTestingMockito.pdf](#).



# Niveluri de testare. Tipuri de bug-uri (1)

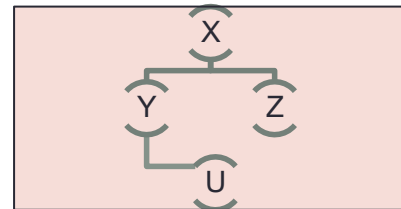
- La testarea unitară și testarea de integrare se pot identifica următoarele tipuri de bug-uri pentru modulele X, Y, Z și U:
  - **la testarea unitară:**
    - bug-uri de funcționalitate în modulele testate:
      - E.g.: **Bug(X)**, **Bug(Y)**, **Bug(Z)**, **Bug(U)**;
  - **la testarea de integrare:**
    - **misuse of interface (MIS):**
      - E.g.: **MIS(X; Y, Z)**, la apelul lui Y și/sau Z din X;
    - **misunderstanding of interface (MUN):**
      - E.g.: **MUN(X; Y, Z)**, la apelul lui Y și/sau Z din X;
    - etc.



# Niveluri de testare. Tipuri de bug-uri (2)

- la testarea unitară:

- bug-uri de funcționalitate în modulele testate:
  - E.g.: **Bug(X)**, **Bug(Y)**, **Bug(Z)**, **Bug(U)**;



## **/\*\* VARIANTA A**

```
* generate and return a list of integer numbers in range [a, b]  
* @param a - lower limit  
* @param b - upper limit  
* @return List<Integer> list of random value in range [a, b]  
*/
```

```
List<Integer> randomNumbers1(int a, int b){  
    int i=0;  
    Random r = new Random();  
    List<Integer> l = new LinkedList<>();  
    while (i<b){  
        l.add(r.nextInt(a));  
        i++;  
    }  
    return l;  
}
```

## **/\*\* VARIANTA B**

```
* generate and return a list of integer numbers in range [a, b]  
* @param a - lower limit  
* @param b - upper limit  
* @return List<Integer> list of random value in range [a, b]  
*/
```

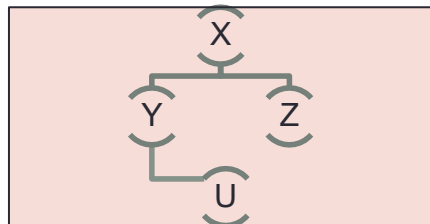
```
List<Integer> randomNumbers(int a, int b){  
    int counter = Math.max((Math.abs(a)), Math.abs(b));  
    Random r = new Random();  
    List<Integer> l = new LinkedList<>();  
    while (counter>0){  
        l.add(r.nextInt(b-a)+a);  
        counter--;  
    }  
    return l;  
}
```

# Niveluri de testare. Tipuri de bug-uri (3)

- la testarea de integrare:

- misuse of interface (MIS):**

- E.g.: MIS(X; Y, Z), la apelul lui Y și/sau Z din X;
- One module makes an error in using the interface of a called module. This is likely to occur in a procedure-call interface. Interface misuse can take the form of **wrong parameter type, wrong parameter order, or wrong number of parameters passed** [Naik, pag.161].*



```
/**
 * generate and return a list of integer numbers in range [a, b]
 * @param a - lower limit
 * @param b - upper limit
 * @return List<Integer> list of random value in range [a, b]
 */
List<Integer> randomNumbers(int a, int b){
    int counter = Math.max((Math.abs(a)), Math.abs(b));
    Random r = new Random();
    List<Integer> l = new LinkedList<>();
    while (counter>0){
        l.add(r.nextInt(b-a)+a);
        counter--;
    }
    return l;
}
```

```
/**
 * generate a list of random values in range [a, b]; return the mean between
 * the minimum and maximum of the generated values
 * @param a - lower limit
 * @param b - upper limit
 * @return the mean between the minimum and maximum of the generated values
 */
double meanRandomList(int a, int b){
    List<Integer> list = randomNumbers(b,a);
    Collections.sort(list);
    return mean(list);
}
```

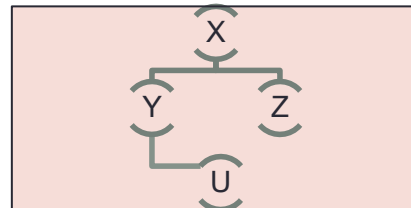
# Niveluri de testare. Tipuri de bug-uri (4)

- la testarea de integrare:

- misunderstanding of interface (MUN):**

- E.g.: MUN(X; Y, Z), la apelul lui Y și/sau Z din X;

• *Misunderstanding of Interface: A calling module may **misunderstand the interface specification of a called module**. The called module may assume that some parameters passed to it satisfy a certain condition, whereas the caller does not ensure that the condition holds. For example, assume that a called module is expected to return the index of an element in an array of integers. The called module may choose to implement binary search with an assumption that the calling module gives it a sorted array. If the caller fails to sort the array before invoking the second module, we will have an instance of interface misunderstanding [Naik, pag.161].*

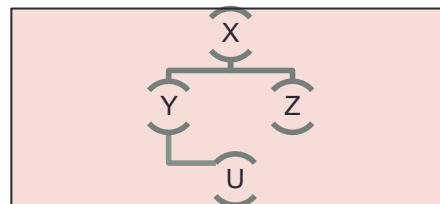


```
/**
 * generate a list of random values in range [a, b]; return the mean between
 * the minimum and maximum of the generated values
 * @param a - lower limit
 * @param b - upper limit
 * @return the mean between the minimum and maximum of the generated
 values
 */
double meanRandomList(int a, int b){
    List<Integer> list = randomNumbers(a,b);
    Collections.sort(list);
    return mean(list);
}
```

```
/**
 * compute the mean between the smallest and the largest values
 * within an ascending ordered list
 * @param list - list of integer number
 * @return the mean value (average)
 */
double mean(List<Integer> list){
    return (double)(list.get(0)+list.get(list.size()-1))/2;
}
```

# Strategii de integrare. Big-bang

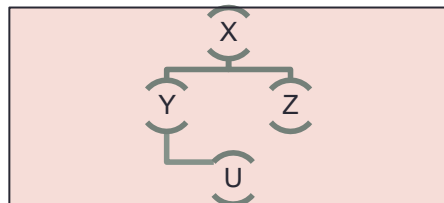
- Integrare non-incrementală: **big-bang**;
  - 1. toate modulele se testează în izolare, i.e., unit testing;
  - 2. se combină simultan toate modulele;



| Nr. crt. | Tip testare | Module     | Driver | Stub/Mock/... | Bug-uri  | Observații  |
|----------|-------------|------------|--------|---------------|--|---|
| a        | Unit        | X          | $D_X$  | $S_Y, S_Z$    | Bug (X)  | - Stub-urile nu au bug-uri<br>- X folosește direct doar Y și Z, nu și U |
| b        | Unit        | Y          | $D_Y$  | $S_U$         | Bug (Y)  | - Stub-urile nu au bug-uri  |
| c        | Unit        | Z          | $D_Z$  | -             | Bug (Z)  |   |
| d        | Unit        | U          | $D_U$  | -             | Bug (U)  | - Pașii a..d se pot realiza în paralel                                  |
| 1        | Integrare   | X, Y, Z, U | $D_X$  | -             | + MIS (X; Y, Z); MUN (X; Y,Z);<br>MIS (Y; U); MUN (Y; U) | - Pot apărea toate tipurile de bug-uri simultan                         |



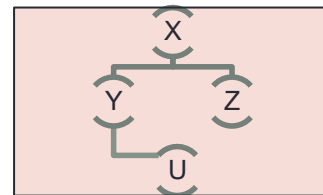
# Strategii de integrare. Top-down



- **Integrare incrementală: top-down, depth first;**
  - 1. se consideră că toate modulele au fost testate în izolare, i.e., unit testing;
  - 2. se realizează integrarea începând cu modulul X, până se ajunge la modulul frunză al unei ramificații; se continuă apoi cu celelalte ramificații, până când toate modulele sunt integrate;

| Nr. crt. | Tip testare | Module     | Driver | Stub/Mock/... | Bug-uri                  | Observații                                  |
|----------|-------------|------------|--------|---------------|--------------------------|---|
| a        | Unit        | X          | $D_X$  | $S_Y, S_Z$    | Bug(X)                   | - Stub-urile nu au bug-uri                  |
| b        | Unit        | Y          | $D_Y$  | $S_U$         | Bug (Y)                  |   |
| c        | Unit        | Z          | $D_Z$  | -             | Bug (Z)                  |   |
| d        | Unit        | U          | $D_U$  | -             | Bug (U)                  |   |
| 1        | Integrare   | X, Y       | $D_X$  | $S_U, S_Z$    | + MIS (X; Y), MUN (X; Y) | + tipurile de bug-uri de la pașii anteriori |
| 2        | Integrare   | X, Y, U    | $D_X$  | $S_Z$         | + MIS (Y; U), MUN (Y; U) |   |
| 3        | Integrare   | X, Y, U, Z | $D_X$  | -             | + MIS (X; Z), MUN (X; Z) |   |

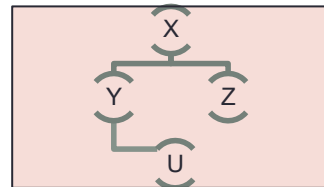
# Strategii de integrare. Top-down



- **Integrare incrementală: top-down, breadth-first;**
  - 1. se consideră că toate modulele au fost testate în izolare, i.e., unit testing;
  - 2. se realizează integrarea începând cu modulul A, integrând toate modulele aflate pe nivelul imediat următor; se continuă apoi cu celelalte niveluri până când toate modulele sunt integrate;

| Nr. crt. | Tip testare | Module     | Driver               | Stub/Mock/...                       | Bug-uri                  | Observații                                  |
|----------|-------------|------------|----------------------|-------------------------------------|--------------------------|---|
| <i>a</i> | <i>Unit</i> | <i>X</i>   | <i>D<sub>X</sub></i> | <i>S<sub>Y</sub>, S<sub>Z</sub></i> | <i>Bug(X)</i>            | - Stub-urile nu au bug-uri                  |
| <i>b</i> | <i>Unit</i> | <i>Y</i>   | <i>D<sub>Y</sub></i> | <i>S<sub>U</sub></i>                | <i>Bug (Y)</i>           |   |
| <i>c</i> | <i>Unit</i> | <i>Z</i>   | <i>D<sub>Z</sub></i> | -                                   | <i>Bug (Z)</i>           |   |
| <i>d</i> | <i>Unit</i> | <i>U</i>   | <i>D<sub>U</sub></i> | -                                   | <i>Bug (U)</i>           |   |
| 1        | Integrare   | X, Y       | D <sub>X</sub>       | S <sub>Z</sub> , S <sub>U</sub>     | + MIS (X; Y), MUN (X; Y) | + tipurile de bug-uri de la pașii anteriori |
| 2        | Integrare   | X, Y, Z    | D <sub>X</sub>       | S <sub>U</sub>                      | + MIS (X; Z), MUN (X; Z) |   |
| 3        | Integrare   | X, Y, Z, U | D <sub>X</sub>       | -                                   | + MIS (Y; U), MUN (Y; U) |   |

# Strategii de integrare. Bottom-up

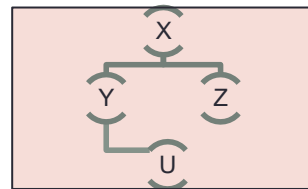


- **Integrare incrementală: bottom-up;**

- 1. se realizează integrarea începând cu modulele frunză, care se testează în izolare, i.e., unit testing;
- 2. se continuă cu integrarea modulelor frunză în modulele de pe nivelurile superioare, până când toate modulele sunt integrate și se obține funcționalitatea completă a sistemului;

| Nr. crt.    | Tip testare | Module     | Driver | Stub       | Bug-uri                  | Observații   |
|-------------|-------------|------------|--------|------------|--------------------------|--|
| a (cluster) | Unit        | Z          | $D_Z$  | -          | Bug (Z)                  | - Se poate realiza în paralel cu b                                     |
| b (cluster) | Unit        | U          | $D_U$  | -          | Bug (U)                  | - Se poate realiza în paralel cu a                                     |
|             | Unit        | Y          | $D_Y$  | $S_U$      | Bug (Y)                  | + tipurile de bug-uri de la pașii anteriori                            |
|             | Integrare   | Y, U       | $D_Y$  | -          | + MIS (Y; U), MUN (Y; U) |  |
| c           | Unit        | X          | $D_X$  | $S_Y, S_Z$ | Bug(X)                   | - Se realizează după a și b<br>- Se vor integra ulterior Y și Z        |
| 1           | Integrare   | X, Y, U    | $D_X$  | $S_Z$      | + MIS (X; Y), MUN (X; Y) | + tipurile de bug-uri de la pașii anteriori<br>- Se poate inversa cu 2 |
| 2           | Integrare   | X, Y, U, Z | $D_X$  | -          | + MIS (X; Z), MUN (X; Z) | + tipurile de bug-uri de la pașii anteriori<br>- Se poate inversa cu 1 |

# Strategii de integrare. Sandwich



- **Integrare mixtă: sandwich;**
  1. se realizează integrarea începând atât cu modulele frunză, cât și cu modulul principal;
  2. pentru modulele din partea superioară a ierarhiei se aplică integrare **top-down**; pentru modulele frunză se aplică integrare **bottom-up**; pentru modulele aflate pe niveluri intermediare se aplică integrarea **big-bang**;

| Nr. crt.                 | Tip testare | Module     | Driver         | Stub                            | Bug-uri                        | Observații   |
|--------------------------|-------------|------------|----------------|---------------------------------|--------------------------------|--|
| a<br>top-down            | Unit        | X          | D <sub>x</sub> | S <sub>y</sub> , S <sub>z</sub> | Bug(X)                         | - Se poate realiza în paralel cu b și c  |
| b (cluster)<br>bottom-up | Unit        | Z          | D <sub>z</sub> | -                               | Bug (Z)                        | - Se poate realiza în paralel cu a și c  |
| c (cluster)<br>bottom-up | Unit        | U          | D <sub>u</sub> | -                               | Bug (U)                        | - Se poate realiza în paralel cu a și b<br><br>+ tipurile de bug-uri de la pașii anteriori |
|                          | Unit        | Y          | D <sub>y</sub> | S <sub>u</sub>                  | Bug (Y)                        |  |
|                          | Integrare   | Y, U       | D <sub>y</sub> | -                               | + MIS (Y; U), MUN (Y; U)       |  |
| 1<br>big-bang            | Integrare   | X, Y, U, Z | D <sub>x</sub> | -                               | + MIS (X; Y, Z), MUN (X; Y, Z) | - Se realizează după a, b și c   |

# Seminar 04. Problemă

- Enunțul din [Seminar04.pdf](#)
- Rezolvare:
  - Echipe: 3-5 studenți;
  - Timp de lucru: 30 minute;
- Task-uri:
  - 1. Elaborarea diagramei de dependență dintre module;
  - 2. Identificarea strategiei de integrare aplicabilă, pe baza contextului de dezvoltare luat în considerare;
  - 3. Elaborarea tabelului cu operațiile de testare unitară și de integrare pentru strategia aplicată (una dintre cele 5 strategii discutate în exemplul anterior);
  - 4. Analiza strategiilor de integrare, i.e., completarea tabelului indicat pentru criteriile indicate (discuție cu toate echipele).