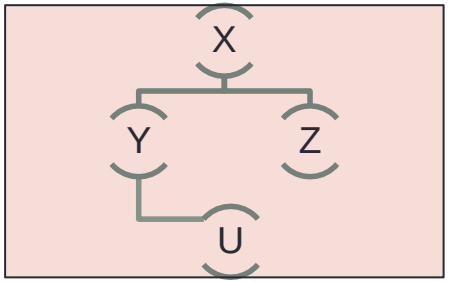
Seminar 04. Continut

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

Niveluri de testare. Exemplu

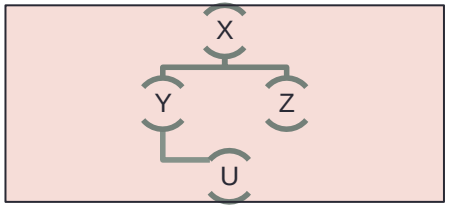
- **Se consideră următoarea diagrama de dependenţă:**



- **Se cere să se simuleze testarea de integrare pentru modulele X, Y, Z şi 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 şi testarea de integrare se consideră următoarele tipuri de module necesare pentru a testa X, Y, Z şi U:

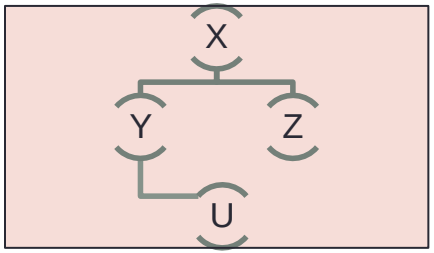


  - **driver** pentru modulele X, Y şi Z – **D_X , D_Y , D_Z , D_U**;
  - **stub** (mock, fake, dummy, spy) pentru modulele Y şi Z – **S_Y , S_Z , S_U.**

- vezi **Lab04_TutorialUnitTestingMockito.pdf.**

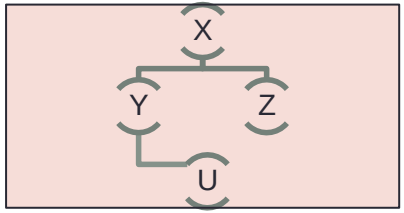
Niveluri de testare. Tipuri de bug-uri (1)

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



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

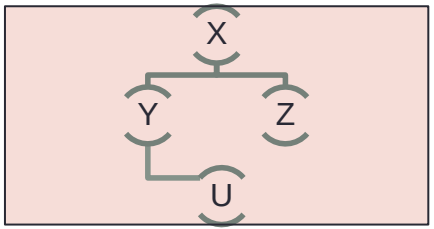
- **la testarea unitară:**
  - bug-uri de funcţionalitate în modulele testate:
    - E.g.: **Bug(X), Bug(Y), Bug(Z), Bug(U)**;




```
/** 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)

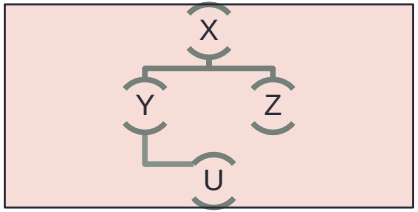


- **la testarea de integrare:**
  - **misuse of interface (MIS):**
    - E.g.: **MIS(X; Y, Z), la apelul lui Y şi/sau Z din X;**
    - ***One module makes an error in using the interface of a called module. This is likely to occur in a procedure–call interface. Interface misuse can take the form of wrong 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)



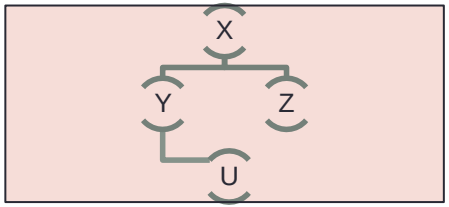
- **la testarea de integrare:**
  - **misunderstanding of interface (MUN):**
    - E.g.: **MUN(X; Y, Z), la apelul lui Y şi/sau Z din X;**

•***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;
}
```

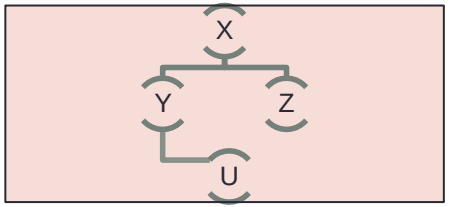
Strategii de integrare. Big-bang



- **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	Observaţii
a	*Unit*	*X*	D_X	S_Y, S_Z	*Bug (X)*	*- Stub-urile nu au bug-uri* *- X foloseşte direct doar Y şi Z, nu şi 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)*	*- Paşii a..d se pot realiza în paralel*
1	**Integrare**	**X, Y, Z, U**	D_X	-	**+ MIS (X; Y, Z); MUN (X; Y,Z); MIS (Y; U); MUN (Y; U)**	**- Pot apărea toate tipurile de bug-uri simultan**

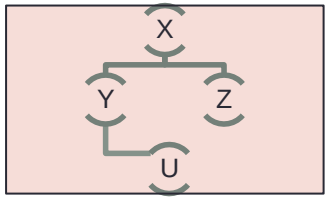
Strategii de integrare. Top-down



- **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 ramificaţii; se continuă apoi cu celelalte ramificaţii, până când toate modulele sunt integrate;

Nr. crt.	Tip testare	Module	Driver	Stub/Mock/...	Bug-uri	Observaţii
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 paşii 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)	

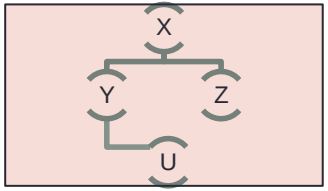
Strategii de integrare. Top-down



- **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	Observaţii
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_Z,S_U	+ MIS (X; Y), MUN (X; Y)	+ tipurile de bug-uri de la paşii anteriori
2	Integrare	X, Y, Z	D_X	S_U	+ MIS (X; Z), MUN (X; Z)	
3	Integrare	X, Y, Z, U	D_X	-	+ MIS (Y; U), MUN (Y; U)	

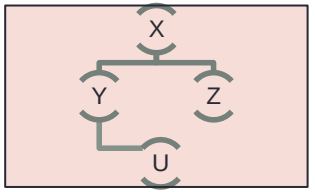
Strategii de integrare. Bottom-up



- **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 şi se obţine funcţionalitatea completă a sistemului;

Nr. crt.	Tip testare	Module	Driver	Stub	Bug-uri	Observaţii
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)*	
	Integrare	Y, U	D_Y	-	+ MIS (Y; U), MUN (Y; U)	*+ tipurile de bug-uri de la paşii anteriori*
c	*Unit*	*X*	D_X	S_Y, S_Z	*Bug(X)*	*- Se realizează după a şi b* *- Se vor integra ulterior Y şi Z*
1	Integrare	X, Y, U	D_X	S_Z	+ MIS (X; Y), MUN (X; Y)	+ tipurile de bug-uri de la paşii anteriori - 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 paşii anteriori - Se poate inversa cu 1

Strategii de integrare. Sandwich



- **Integrare mixtă: sandwich;**
  - 1. se realizează integrarea începând atât cu modulele frunză, cât şi 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	Observaţii
a top-down	Unit	X	D_X	S_Y, S_Z	Bug(X)	- Se poate realiza în paralel cu b şi c
b (cluster) bottom-up	Unit	Z	D_Z	-	Bug (Z)	- Se poate realiza în paralel cu a şi c
c (cluster) bottom-up	Unit	U	D_U	-	Bug (U)	- Se poate realiza în paralel cu a şi b
	Unit	Y	D_Y	S_U	Bug (Y)	
	Integrare	Y, U	D_Y	-	+ MIS (Y; U), MUN (Y; U)	+ tipurile de bug-uri de la paşii anteriori
1 big-bang	Integrare	X, Y, U, Z	D_X	-	+ MIS (X; Y, Z), MUN (X; Y, Z)	- Se realizează după a, b şi c

Seminar 04. Problemă

- **Enunţul din Seminar04.pdf**
- **Rezolvare:**
  - **Echipe: 3-5 studenţi;**
  - **Timp de lucru: 30 minute, în timpul seminarului;**
- **Task-uri:**
  - **1. Formarea echipelor + alegerea unui context (C1..C5);**
    - **Se va completa într-un fişier pe Google Drive numele şi componenţa echipei şi contextul ales;**
    - **Se va avea în vedere analizarea fiecărui context de către cel puţin o echipă;**
  - **2. Elaborarea diagramei de dependenţă dintre module;**
  - **3. Elaborarea tabelului cu operaţiile de testare unitară şi de integrare pe baza contextului ales, indicând strategia de integrare adecvată (una dintre cele discutate în exemplul anterior);**
    - **Se va transmite prin Skype un fişier cu rezolvarea cerinţelor 2. şi 3.;**
    - **Pentru cerinţele 2. şi 3. echipele vor primi cate 2 puncte de activitate la seminarul 4.**