

Софтверско прашања прв дел и втор дел

1. What does the “3-year half-life” in software development knowledge mean? / Што означува правилото (получивот од 3 години) во знаењата за развој на софтвер?

Select one:

- a.) Playing Half – life for 3 years will make you better software engineer / играњето на Half – life во времетраење од 3 години ќе направи подобар софтвер инженер. - не е tochno
b.) In 3 years you will know half of what you know today / за 3 години ќе знаете пола од тоа што го знаете денес - не е tochno

- c.) Half of what you need to know today will be obsolete within 3 years/ Пона од тоа што треба да го знаете денес ќе биде застарено за 3 години - OVA Е TOCHNO
d.) Every three years as a software engineer your life is shortened by one half / Секои три години како софтверски инженер ви го кратат животниот век за половина. - не е tochno

What does the “3-year half-life” in software development knowledge mean? // Што означува правилото “3-year half-life” („получивот“ од 3 години) во знаењата за развој на софтвер?

Select one:

- a. playing Half-life for 3 years will make you better software engineer // играњето на Half-life во времетраење од три години ќе направи подобар софтвер инженер
- b. in 3 years you will know half of what you know today // за 3 години ќе знаете пола од тоа што го знаете денес
- c. half of what you need to know today will be obsolete within 3 years // пола од тоа што треба да го знаете денес ќе биде застарено за 3 години
- d. every three years as a software engineer your life is shortened by one half // секои три години како софтверски инженер ви го кратат животниот век за половина

(slika za prashanje 1)

Tochno e pod c. - 100%

2. Кои од наведените работи се точни за Gradle. / Select the correct statements about Gradle
Select one or more:

- a.) само програмерот што го развива софтверот може да има увид во резултатите од градењето на софтверот / only the programmer that develops the software has insight into the results of the software build.-не е tochno
b.) даден проект во Gradle може да се изврши и без да имаме инсталирало Gradle на машината / a Gradle project can be run without having Gradle installed on the machine. - tochno е.

*Objasnuvanje Viktor: moze da se izvrshi i bez da instaliras Gradle ako vo projektot ima Gradle Wrapper

- c.) дозволува да се вметнуваат нови задачи (tasks) / allows to add new tasks - tochno
d.) овозможува пишување и извршување на тестови / enables writing and running of tests - не е tochno - Се извршуваат, но не се пишуваат.

e.) намената е исклучиво за проекти коишто чијшто изворен код е во Java /
It's purpose is only for projects who are written in Java - ova ne e tochno, beshe
одговорено како tochno, исто коментарот на колегата viktor е точен исто... OVA NE E SAMO ZA
JAVA, може и kotlin или dr jazik!

*Objasnuvanje Viktor: Мислам дека не е самоа проекти чијшто изворен код е во Java.

Gradle is a build automation tool often used for JVM languages such as Java, Groovy or Scala.

Source: Google

f.) нуди интеграција со голем број на интегрирани околини за развој (IDE) / Offers integration with a lot of IDEs - tochno e
g.) алатка која служи само за да се компајлира и изврши даден код / a tool that serves only for compilation and execution of a code - ne e tochno - ne e samo za toa...

Koi od navedenite работи се точни за Gradle. // Select the correct statements about Gradle

Select one or more:

- a. овозможува пишување и извршување на тестови // enables writing and running of tests
- b. даден проект во Gradle може да се изврши и без да имаме инсталерирано Gradle на машината // a Gradle project can be run without having Gradle installed on the machine
- c. дозволува да се вметнуваат нови задачи (tasks) // allows to add new tasks
- d. само програмерот што го развива софтверот може да има увид во резултатите од градењето на софтверот // only the programmer that develops the software has insight into the results of the software build
- e. алатка која што служи само за да се компајлира и изврши даден код // a tool that serves only for compilation and execution of a code
- f. нуди интеграција со голем број на интегрирани околини за развој (IDE) // offers integration with a lot of IDEs
- g. наменета е исклучиво за проекти коишто чијшто изворен код е во Java // it's purpose is only for projects who are written in Java

(slika za 2ro prashanje)

Tochno e pod b, c i f.

Gradle is a build automation tool for multi-language software development. It controls the development process in the tasks of compilation and packaging to testing, deployment, and publishing.

ne e samo za buildanje i izvrsuvanje kod

moze i instalacii da pravi

primer instaliri gi tie i tie biblioteki pred build

drugo so moze da mu kazas e deployni go buildot na AWS ili heroku etc,

3. Legacy system are usually critical to the business in which they operate, thus their business value should be assessed. Determine the most suitable issues of legacy system with low business value. // Наследените системи се обично критични за бизнисите во коишто оперираат, така што треба да се процени нивната деловна вредност. Определете ги најпогодните проблеми со таквите системи со ниска деловна вредност.

Select one or more:

- a.) Systems with frequent use or systems used by many clients / Системи коишто се користат често или системи коишто ги користат голем број Клиенти. - не е точно
- b.) Systems with occasional use or systems used by few clients / системи што се користат повремено или системи што се користат од мал број клиенти. - точно е
- c.) Systems that force the use of inefficient business processes / Системи коишто налагаат употреба на неефикасни бизнис процеси. - точно е
- d.) Systems that depend on system outputs / Системи што зависат од излезот од системот - не е точно

Legacy systems are usually critical to the business in which they operate, thus their business value should be assessed. Determine the most suitable issues of legacy systems with low business value. // Наследените системи се обично критични за бизнисите во коишто оперираат, така што треба да се процени нивната деловна вредност. Определете ги најпогодните проблеми со таквите системи со ниска деловна вредност.

Select one or more:

- a. Systems with frequent use or systems used by many clients // Системи коишто се користат често или системи коишто ги користат голем број клиенти
- b. Systems that force the use of inefficient business processes // Системи коишто налагаат употреба на неефикасни бизнис процеси
- c. Systems with occasional use or systems used by few clients // Системи што се користат повремено или системи што се користат од мал број клиенти
- d. Systems that depend on system outputs // Системи што зависат од излезот од системот

(slika za 3to prashanje)

Tochni se b i c.

Нормално одржување на системот.

14. Проценка на бизнис вредноста – проценката треба да земе предвид различни гледишта од страна на крајните корисници, клиенитите, линиските менаџери, ИТ менаџери, раководители, интервју со стекхолдери итн.

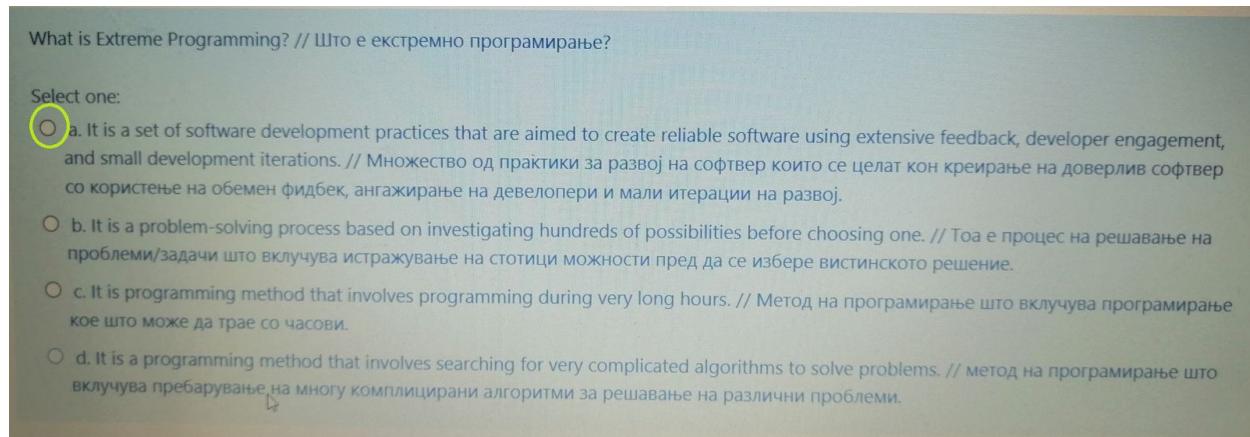
-Issues:

1. Искористеноста на системот – ако системите се користат повремено, или од мал број на луѓе, тие имаат мала бизнис вредност
2. Поддржаните бизнис процеси – мала бизнис вредност, ако принудува употреба на неефикасни бизнис процеси
3. Сигурност на системот – ако системот не е сигурен и проблемите директно влијаат на клиентите, мала бизнис вредност
4. Излезите на системот – ако бизнисот зависи од аутпутите на системот, тогаш тој систем има висока бизнис вредност.

4. What is Extreme Programming / Што е екстремно програмирање?

Select one:

- a.) It is a set of software development practices that are aimed to create reliable software using extensive feedback developer engagement, and small development iterations. // множество од практики за развој на софтвер коишто се целат кон креирање на доверлив софтвер со користење на обемен фидбек, ангажирање на девелопери, и мали итерации на развој. - точно е
- b.) It is a problem - solving process based on investigating hundreds of possibilities before choosing one. // Тоа е процес на решавање на проблеми/задачи што вклучуваат истражување на стотици можности пред да се избере вистинското решение. - не е точно
- c.) it is programming method that involves programming during very long hours // Метод на програмирање што вклучува програмирање кое може да трае со часови. - не е точно
- d.) It is a programming method that involves searching for very complicated algorithms to solve problems. // метод на програмирање што вклучува пребарување на многу комплицирани алгоритми за решавање на различни проблеми - не е точно



(slika za 4to prashanje)

Tochno e pod a.

Extreme programming is a software development methodology which is intended to improve software quality and responsiveness to changing customer requirements.

5. Component qualification ensures: / Квалификацијата на компоненти гарантира:

Select one:

- a.) дека компонента-кандидат ќе ги исполнi сите побарани критериуми за перформанси на апликацијата - не е точно
- b.) дека компонента-кандидат ќе ги исполнi сите побарани критериуми за Апликацијата - не е точно
- c.) дека компонента-кандидат ќе ги изврши функциите коишто се потребни за комуникација со надворешен систем. - не е точно
- d.) дека компонента - кандидат ќе ги изврши функциите коишто се потребни за апликацијата - точно е

Component qualification ensures: // Квалификацијата на компоненти гарантира:

Select one:

- a. that a candidate component will fulfill all design criterias required for the application // дека компонента-кандидат ќе ги исполнi сите побарани критериуми за апликацијата
- b. that a candidate component will fulfil all performance criterias required for the application // дека компонента-кандидат ќе ги исполнi сите побарани критериуми за перформанси на апликацијата
- c. that a candidate component will perform the function required for the external system communication // дека компонента-кандидат ќе изврши функциите коишто се потребни за комуникација со надворешен систем
- d. that a candidate component will perform the function required for the application // дека компонента-кандидат ќе ги изврши функциите коишто се потребни за апликацијата

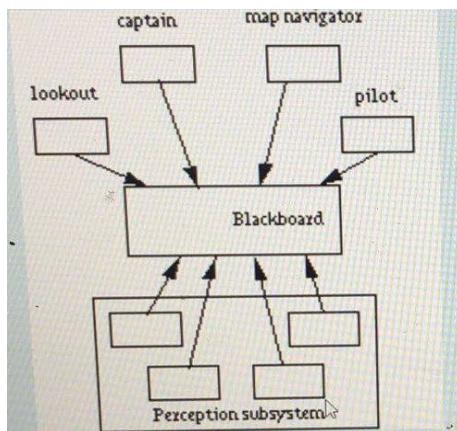
(slika za prashanje 5)

Tochno e pod d.

Component Qualification

- Component qualification ensures that a candidate component will perform the function required, will properly “fit” into the architectural style specified for the system, and will exhibit the quality characteristics (e.g., performance, reliability, usability) that are required for the application.

6. What architectural pattern is shown on this picture? / Кој архитектонски шаблон е претставен на слика?



(слика за бто прашање)

- a.) Архитектура со репозиториум (BLACKBOARD е репото) - tochno e
- b.) Слоевита архитектура

- c.) Pipe and filter architecture
- d.) Client - server architecture

Податочно централизирана архитектура

Карактеристично за овој модел е што податоците се централизирани (сместени се на еден потсистем), и до нив пристапуваат останатите компоненти на кои им се потребни тие податоци. Компонентите кои пристапуваат до заедничка структура на податоци и се релативно независни, што овозможува промена на самата структура, односно додавање на нова компонента или исфрлување на веќе постоечка компонента, без да се наруши работата на целиот систем. Не постои директна комуникација помеѓу компонентите, туку тие меѓусебно комуницираат само преку заедничката база на податоци.

{ 44 }

Преизведен од gl.vi.ne <25 Sep 2020 - 03:17>



Слика 23. Податочно централизирана архитектура

Најпознат пример за ваков тип на архитектурата е архитектурата на база на податоци, кај која се користи заедничка шема на податоците со протокол за дефинирање на податоци (на пример, збир на поврзани табели со полиња и типови на податоци во RDBMS).

Кај овој архитектурен стил, разликуваме две категории, во зависност од протоколот на контрола, и тоа:

- Репозиториум (*Repository Architecture Style*)
- Blackboard Architecture Style

Кај репозиториумот, складиштето на податоци е пасивно, додека клиентите (софтверски компоненти) се активни и тие го контролираат протокот на логиката. Клиентот испраќа барање до системот за да изврши одредено действие (пр. внесување на податоци, ажурирање на податоци и сл.). Компјутерските процеси се независни и предизвикани од влезните барања (слика 24).

Предностите кај овој тип на архитектура се тоа што се обезбедува интегритет на податоците, како и приспособливост и повторна употреба на компонентите. Додека, негативностите пред се се поврзани со податочната структура на складиштето на податоци. Клиентите треба да се покоруват на заеднички модел на податоци, што може лошо да се одрази на перформансите на системот во целина; која било промена на структурата на податоци влијае многу и врз клиентите; еволуцијата на податоците е тешка и скапа; трошок за пренесување на податоците на мрежа за дистрибуирани податоци е голем.



Кај архитектурниот стил *Blackboard*, складиштето на податоци е активно, додека клиентите се пасивни. Па така, логичниот проток се одредува спрема моменталниот статус на податоците во складиштето на податоци. Кај овој стил постои една компонента наречена *blackboard* (црна табла), која делува како централна база на податоци, преку која се врши комуникација помеѓу клиентите. Складиштето на податоци ги известува

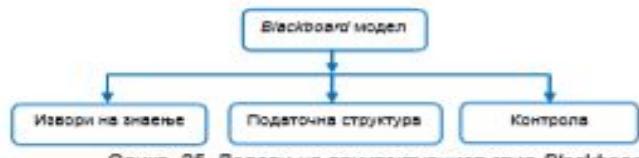
клиентите секогаш кога има промени во податоците. Тоа испраќа известување познато како активирач (*trigger*), заедно со податоците до клиентите. Овој пристап често е користен во одредени комплексни апликации, како што се препознавање на говор, препознавање на слики, безбедносни системи и системи за управување со деловните ресурси итн.

Архитектурниот стил *Blackboard* обично е претставен со три главни дела (слика 25).

Извори на знаење (*The knowledge sources*) - познати и како слушатели (*Listeners*) или претплатници (*Subscribers*), кои претставуваат различни и независни единици. Тие решаваат одредени делови од проблемот и ги обединуваат делумните резултати. Интеракцијата меѓу изворите на знаење се одвива единствено преку *blackboard*.

Податочна структура - податоците за решавање на проблемите се организирани во хиерархија зависна од апликацијата. Извори на знаење прават промени во *blackboard* што постепено доведува до решение за проблемот.

Контрола - таа ги управува задачите и ја проверува работната состојба.

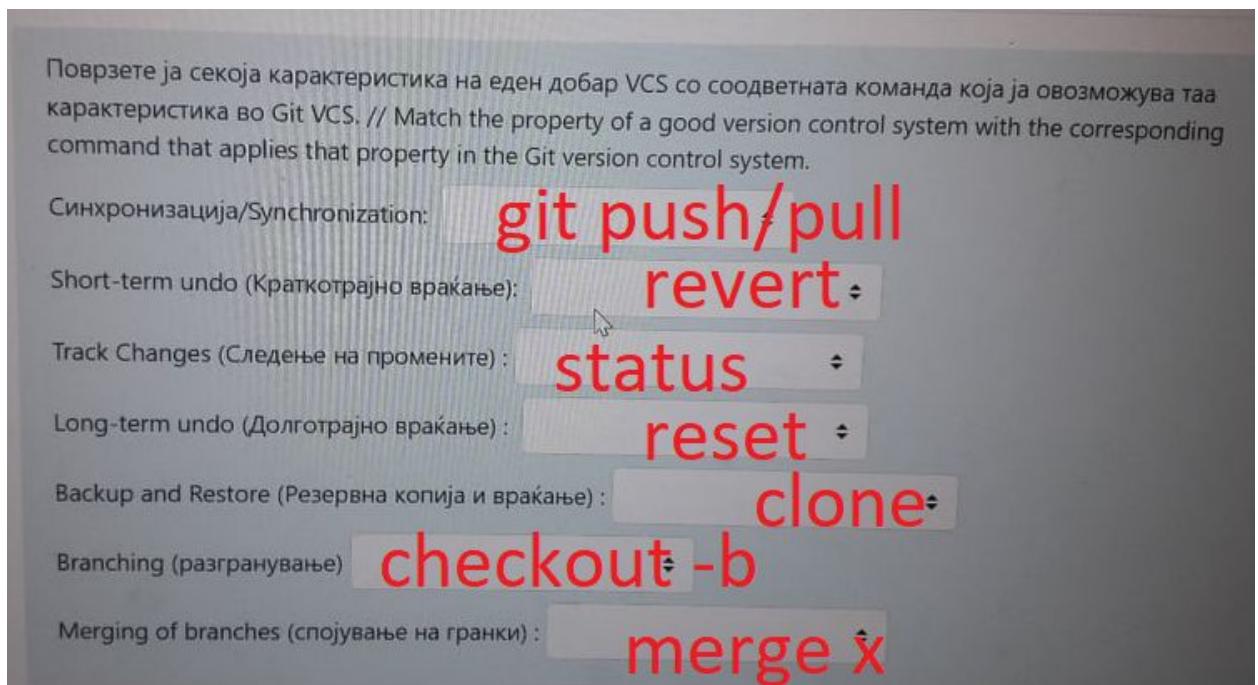


Предностите кај овој тип на архитектура се: конкурентноста - која им овозможува на сите извори на знаење да работат паралелно, независно еден од друг; приспособливост – може лесно да се добаваат или ажурираат изворите на знаење; поддржува повторна употребливост на изворите на знаење.

Како и кај претходниот модел, и кај моделот *Blackboard*, структурната промена на податоците во *blackboard* може да има значително влијание врз сите компоненти. Понатаму, овој модел има одредени проблеми при синхронизација на повеќе агенти, па затоа дизајнирањето и тестирањето на системот честопати претставува предизвик.

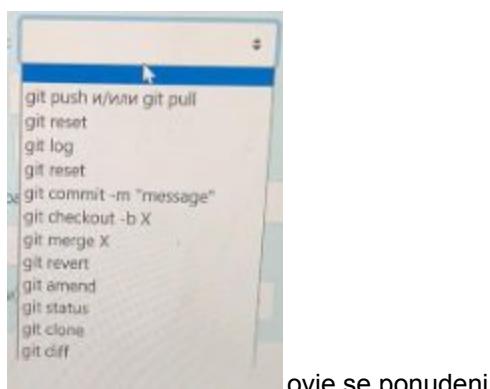
7. Поврзете ја секоја карактеристика на еден добар VCS со соодветната команда која ја овозможува таа карактеристика во GIT VCS

Синхронизација git pull/git push
Short - term undo (краткотрајно враќање) revert
Track Changes (Следење на промените) git log или git status
Long - term undo (долготрајно враќање) reset
Backup and Restore (резервна копија и враќање) git clone
Branching (разгранување) checkout -b
Merging of branches (спојување на гранки) merge x



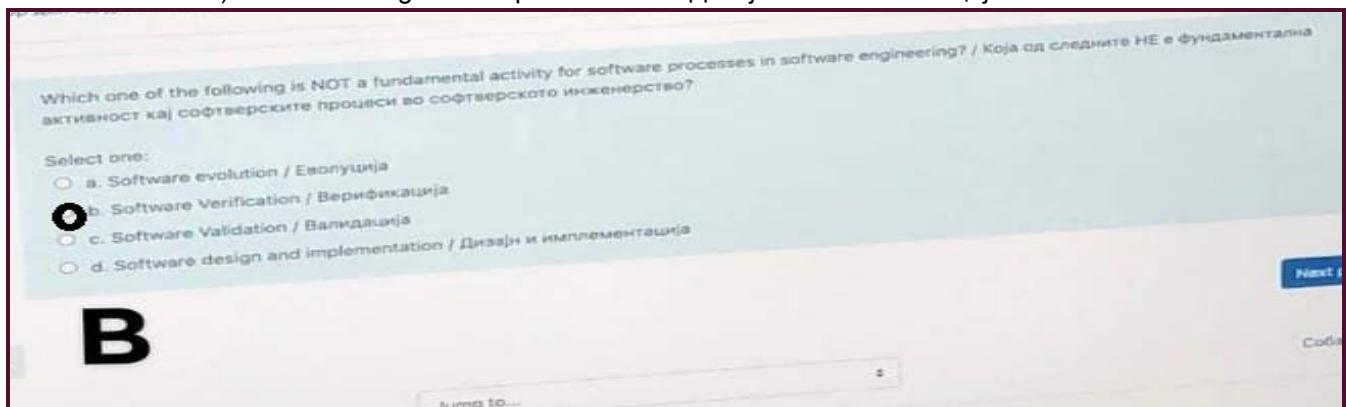
(slika za 7mo prashanje)

(OD koleshkata vo grupa - ubavo napishano)



moze i so revert i so reset
zavisi od test slucajot,
ako promenite ti se pushnati, revert e po soodvetna komanda, ke kreira revert commit i okej si e
ako promenite ti se lokalno, moze i so reset --soft--hard
ako se pushnati promenite i go pravis so reset, ke treba force push da napravis, a nekoj ako ima
pullnato, ke izbrishe cela istorija
git reset - - soft HEAD^
undo last commit put changes into staging

8. Which one of the following is NOT a fundamental activity for software processes in software engineering? / Која од следните НЕ е фундаментална активност кај софтверските процеси во софтверското инженерство?
- Select one:
- a.) Software evolution / Еволуција
 - b.) Software Verification / Верификација - ова не е fundamentalna**
 - c.) Software Validation / Валидација
 - d.) Software design and implementation / Дизајн и имплементација



(слика за 8мо прашање)

- Главни процесни активности се: Спецификација, развој (дизајн и имплементација), валидација, еволуција. (ДОКОЛКУ ИМА ОБРАТНО ПРАШАЊЕ).
9. Which of the following is NOT true? / Што од наведеното НЕ Е точно?
- Select one:
- a.) Валидацијата е процес на евалуација на софтверот со цел да се утврди дали ги задоволува барањата на клиентот. - tochno e**
 - b.) Тестирањето го спроведува инженерот кој го развива софтверот или независна група за тестирање. - ова е tochno**
 - c.) Регресиско тестирање е повторното извршување на подмножество тестови што се веќе спроведени со цел да се обезбеди дека софтверот правилно имплементира специфична функција. - provereno, ова е netochno**
 - d.) Тестирањето и дебагирање се различни активности, но дебагирање мора да биде дел од секоја стратегија за тестирање. - tochno e**

*Filip:

Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

Which of the following is NOT true? // Што од наведеното НЕ е точно?

Select one:

- a. Validation is the process of evaluating software to determine whether it satisfies specified customer requirements. // Валидацијата е процес на оценувања на софтверот со цел да се утврди дали ги задоволува барањата на клиентот.
- b. Testing is conducted by the developer of the software or an independent test group. // Тестирањето го спроведува инженерот кој го развија софтверот или независна група за тестирање.
- c. Regression testing is the re-execution of some subset of tests that have already been conducted to ensure that software correctly implements a specific function. // Регресиско тестирање е повторното извршување на подмножество тестови што се веќе спроведени со цел да се обезбеди дека софтверот правилно имплементира специфична функција.
- d. Testing and debugging are different activities, but debugging must be accommodated in any testing strategy. // Тестирањето и дебагирање се различни активности, но дебагирање мора да биде дел од секоја стратегија за тестирање.

(слика за 9то прашање)

- Регресиско тестирање е повторно извршување на некои подгрупи тестови кои биле спроведени.
- Регресивно тестирање...

5. Регресивно тестирање – повторно извршување на подмножество тестови што се веќе спроведени за да се обезбеди дека промените не предизвикале несакани ефекти.
-Регресивното тестирање помага да се осигураме дека промените не доведуваат ненамерно однесување или пак дополнителни грешки
-Може да се спроведе мануелно, со повторно извршување на подмножеството на сите тест случајеви или пак автоматско со користење на алатки за снимање/репродукција.

10. Black - box wrapping adaptation technique performs? / Техниката на адаптирање Black - box wrapping изведува?

Select one or more:

- a.) промена на API-та на компонентите согласно нашите промени(za grey box)
- b.) пост-процесирање на податоците - **tochno e**
- c.) препроцесирање на податоците - **tochno e**
- d.) промена на внатрешниот изворен код на компонентата согласно нашите потреби(ова е za white box)

Black-box wrapping adaptation technique performs? // Техниката на адаптирање Black-box wrapping изведува?

Select one or more:

- a. change of the APIs of the component according to our needs // промена на API-те на компонентите согласно нашите промени
- b. post-processing of the data // пост-процесирање на податоците
- c. pre-processing of the data // препроцесирање на податоците
- d. change of the internal code of the component according to our needs // промена на внатрешниот изворен код на компонентата согласно нашите потреби

B,C

Собр. 1 ▶

(слика за прашање 10)

11. Нека е дадена класата “CFG” со методот “public static String greeting(int hour)”, каде врз основа на внесениот час се враќа соодветна порака со поздрав. За истиот метод е даден и Control Flow Graph.

Кој е минималниот број на тестови кои треба да се напишат за да се исполнит условот за:

-C0- Every statement методата

2

*Objasnuvanje Viktor:

Dovolni se 2 sluchai vo edniot hour = 9 a vo drugiot hour = 15.

U prviot slucaj ne se opfakjat samo 4 i 5 linija a u vtoriot se opfakjat 4 i 5.

0

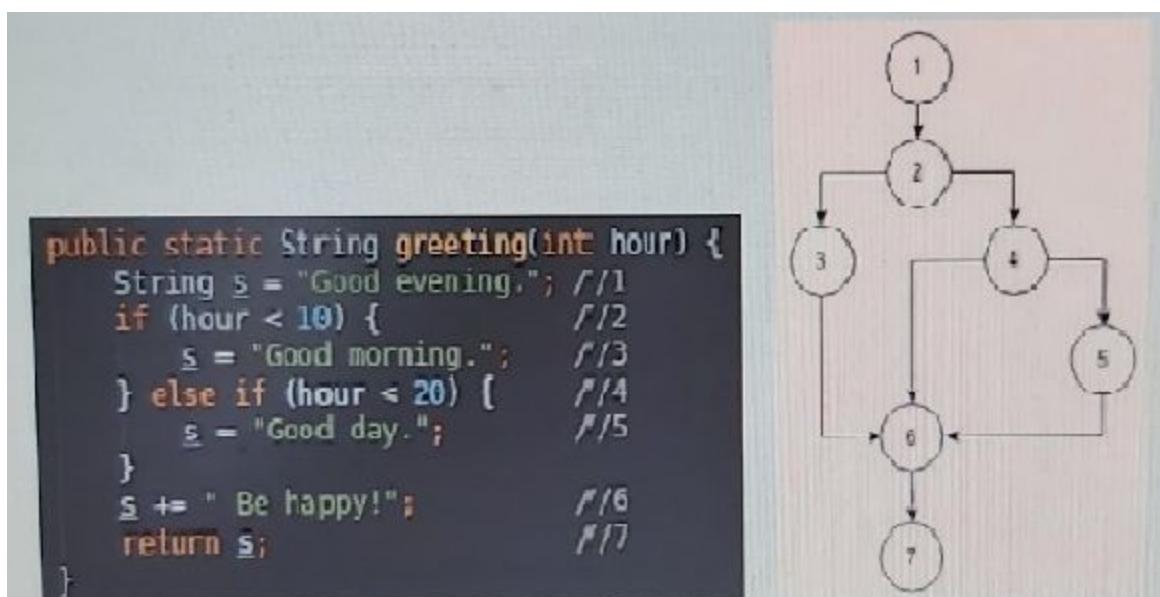
-C1- Every branch методата

3

Koja е цикломатската комплексност?

3

Доколку сакаме да напишеме метод за тестирање на патеката 1-2-3-6-7 како би изгледала тест класата?



(слика за 11-to prashanje)

Училочнска компонентност / CC

Број на јазли $V = 6$
Број на ребра $E = 8$

$$CC = E - V + 1 = 8 - 6 + 1 = 3$$

greshno,
ama
tochno
reshenie

```

String = "Good Evening."
if (hour < 10) {
    s = "Good Morning";
} else if (hour < 20) {
    s = "Good day";
}
s += "Be happy!"
return s;
  
```

1. Every statement

Statements	9	11
1	*	*
2	*	*
3	*	
4	*	*
5		*
6	*	*
7	*	*

2 test cases!

2. Every branch

	9	11	21
1-2	*	*	*
2-3	*		
2-4		*	*
3-6	*		
4-5		*	
4-6		*	
5-6	*		
6-7	*	*	*

3 test cases!

$CC = 2$
 $V = 7$
 $E = 8$

$$CC = 8 - 7 + 2 = 3$$

(OD koleshkata vo grupa - ubavo napishano)

Greshka vo formulata i vo brojot na rebra i jazli!

jazli=7

rebra=8

REBRA-JAZLI+2=3

8-7+2=1+2=3

ILI na drug nachin broime regioni i pak se 3

Greshka vo formulata i vo brojot na rebra!

jazli=7

rebra=8

REBRA-JAZLI+2=3

8-7+2=1+2=3

ILI na drug nachin broime regioni i pak se 3

Одговорете на следните прашања: / Answer the following questions:

Кој е минималниот број на тестови кои треба да се напишат за да се исполнi условот за:
/ What is the minimum number of test cases that need to be written to meet the requirement for:

- C0 - Every statement методата / C0 - Every statement method
- C1 - Every branch методата / C1 - Every branch method

Која е цикломатската комплексност? / What is the cyclomatic complexity?

Доколку сакаме да напишеме метод за тестирање на патеката 1-2-4-5-6-7, како би изгледала тест класата?
/ If we want to write a method for testing the path 1-2-4-5-6-7, what would the test class look like?

```
public class TestClass {  
  
    void greetingTest() {  
        assertEquals("Hello", "Hello");  
    }  
}
```

[Next page](#)

Доколку сакаме да напишеме метод за тестирање на патеката 1-2-4-5-6-7, како би изгледала тест класата?
/ If we want to write a method for testing the path 1-2-4-5-6-7, what would the test class look like?

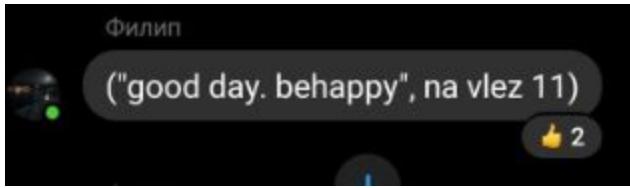
nishto

```
public class TestClass {  
  
    @Test  
    void greetingTest() {  
        assertEquals("Hello", "Hello");  
    }  
}
```

od 10 do 19

nekoe

Od 10 Do 19*



12. Нека е дадена класата “CFG” со методот “public static String getNumberType(int number)”, кој враќа порака дали дадениот број е позитивен, негативен или нула.

За истиот метод е даден и Control Flow Graph.

Кој е минималниот број на тестови кои треба да се напишат за да се исполнi условот за:

-C0- Every statement методата (site jazlii, site gi pominuvame) - site linii

2 (gi pominuvash site kako jazli, sekoja linija e statement, od grafot gledame)

-C1- Every branch методата (celta e da gi pokriesh site mozhni uslovi vo kodot, od onie shto se generira nova granka) - site pateki

3 (site parovi se razgleduваат)

Која е цикломатската комплексност?

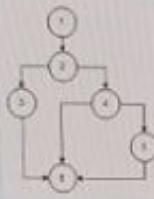
3 (број на региони) E(rebra)-V(jazli)+2

Доколку сакаме да напишеме метод за тестирање на патеката 1-2-3-6 како би изгледала тест класата?

Дадена е класата „CFG“ со методот „public static String getNumberType(int number)“, кој врака порака дали дадениот број е позитивен, негативен или нула. За истиот метод е даден и Control Flow Graph.

/
There is a class "CFG" with a method "public static String getNumberType(int number)", which returns a message whether the given number is positive, negative or zero. For the same method, there is a Control Flow Graph.

```
public static String getNumberType(int number) {
    String s = "Zero";
    if (number > 0) {
        s = "positive";
    } else if (number < 0) {
        s = "negative";
    }
    return s;
}
```



Одговорете на следните прашања / Answer the following questions:

Кој е минималниот број на тестови кои треба да се напишат за да се исполнат условите за:
/ What is the minimum number of test cases that need to be written to meet the requirement for:

- C0 - Every statement методата / C0 - Every statement method

- C1 - Every branch методата / C1 - Every branch method

Која е цикличката компликсност? / What is the cyclomatic complexity?

Доколку сакаме да напишеме метод за тестирање на патеката 1-2-4-6, како би изгледала тест класата?
/ If we want to write a method for testing the path 1-2-4-6, what would the test class look like?

```
public class TestClass {
```

Дадена е класата „CFG“ со методот „public static String getNumberType(int number)“, кој врака порака дали дадениот број е позитивен, негативен или нула. За истиот метод е даден и Control Flow Graph.

/
There is a class "CFG" with a method "public static String getNumberType(int number)", which returns a message whether the given number is positive, negative or zero. For the same method, there is a Control Flow Graph.

```
public static String getNumberType(int number) {
    String s = "Zero";
    if (number > 0) {
        s = "positive";
    } else if (number < 0) {
        s = "negative";
    }
    return s;
}
```

1236
12456

nemora site vo 1
test case, ako gi
pomine moze, ama
nemora

1236

12456

Одговорете на следните прашања / Answer the following questions:

Кој е минималниот број на тестови кои треба да се напишат за да се исполнат условите за:
/ What is the minimum number of test cases that need to be written to meet the requirement for:

- C0 - Every statement методата / C0 - Every statement method

- C1 - Every branch методата / C1 - Every branch method

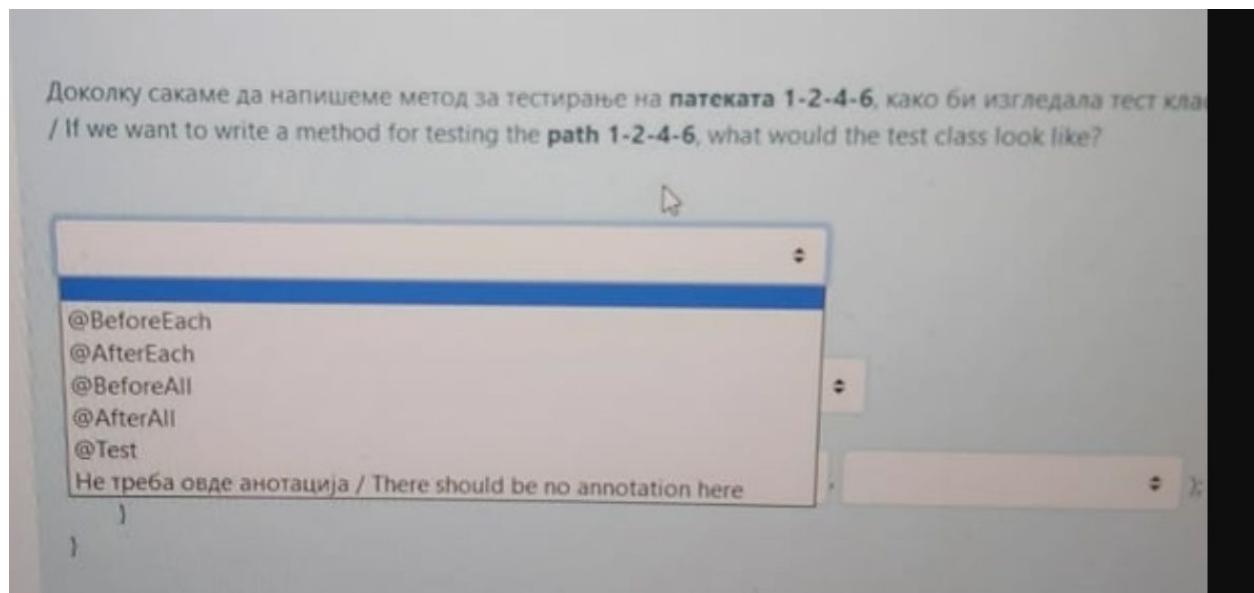
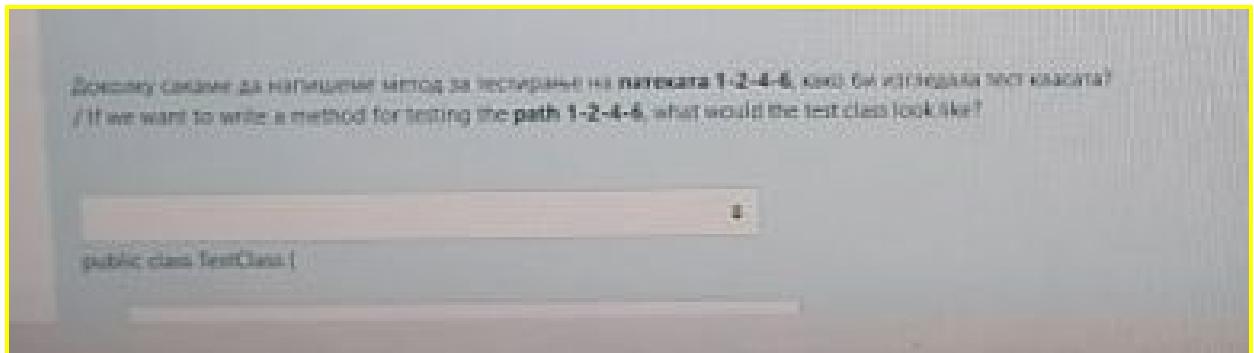
Која е цикличката компликсност? / What is the cyclomatic complexity?

Доколку сакаме да напишеме метод за тестирање на патеката 1-2-4-6, како би изгледала тест класата?
/ If we want to write a method for testing the path 1-2-4-6 what would the test class look like?

```
public class TestClass {
```

	every branch	parovi	1 test	2 test	3 test
SAM	1,2	*	*		
proveruvash za pozitiven broj, pa za negativen broj	2,3	*			
	2,4		*		
	3,6	*			
zimash za 1 za -1 primer	4,5		*		
VO PAROVI	4,6			*	
1 test: 1 zimame 2 test: -1 zimash 3 test: 0 zemi	5,6		*		

every path
se site mozhni patovi...
3vкупно bi imalo
1236
1246
12456



("good day.behappy", greeting(11))

(slika za 12-to prashanje)

Za 1 izbor treba: Ne treba ovde anotacija

2 izbor: @Test

Зависи од барањето, ако не бара ништо, не треба овде анотација.

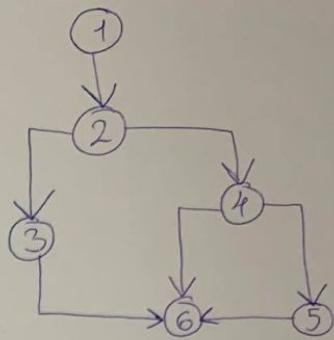
@Test се става пред void.

- + информацији : every path: секој можен пат до завршување на кодот и ги има такви многу
- За тестирање : со assert метод

```

6d. String s = "zero"; //1
if (number > 0) { //2
    s = "positive"; //3
}
else if (number < 0) { //4
    s = "negative"; //5
}
return s; //6

```



1. Every Statement

Statements Test1 Test2

	*	*
1	*	*
2	*	*
3	*	
4		*
5		*
6	*	*

За Test 1 то значе бројот 1.

Сигајки бројот е положителен

и вредноста може 1, 2, 3, 6.

За Test 2 то значе бројот -1.

Сигајки бројот е неиздаден

и вредноста може 1, 2, 4, 5, 6
-јо вредноста може да е 2 а кога тога
тога не е > 0 не вредноста бидејќи 3.

* 3a Every Statement : 2 test cases.

2. Every branch

	T1	T2	T3
1-2	*	*	*
2-3	*		
2-4		*	*
3-6	*		
4-5		*	
4-6			*
5-6	*		

За T1 то значе бројот 1.

За T2 то значе бројот -1.

За T3 то значе бројот 0

* 3a every branch : 3 test cases

3. Every path

1, 2, 3, 6 \rightarrow 1

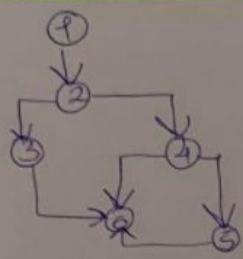
1, 2, 4, 6 \rightarrow 0

1, 2, 4, 5, 6 \rightarrow -1

* 3a every path : 3 test cases

4. За дозволена $1-2-3-6$ може со само кој
од издаден број оп. 1. \uparrow 1 test case

Униконичка конолексија. СС



Број на јазли $V = 6$

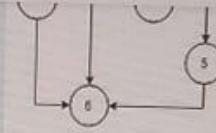
Број на ребра $E = 7$

$$CC = E - V + 1 = 7 - 6 + 1 = 3$$

(OD koleshkata vo grupa - ubavo napisano)

of 13)

```
    s = "Positive";           //3
} else if (number < 0) {      //4
    s = "Negative";          //5
}
return s;                   //6
```



Одговорете на следните прашања: / Answer the following questions:

Кој е минималниот број на тестови кои треба да се напишат за да се исполнит условот за:
/ What is the minimum number of test cases that need to be written to meet the requirement for:

- C0 - Every statement методата / C0 - Every statement method

- C1 - Every branch методата / C1 - Every branch method

Која е цикломатската комплексност? / What is the cyclomatic complexity?

Доколку сакаме да напишеме метод за тестирање на **патеката 1-2-4-6**, како би изгледала тест класата?
/ If we want to write a method for testing the **path 1-2-4-6**, what would the test class look like?

```
public class TestClass {
```

```
    void getNumberTypeTest() {
```

```
        assertEquals(
```

```
    }
```

```
CFG.getNumberType(3)
CFG.getNumberType(-3)
CFG.getNumberType(0)
```

NE TREBA

Доколку сакаме да напишеме метод за тестирање на патеката 1-2-4-5; како би изгледала тест класата?
 / If we want to write a method for testing the path 1-2-4-5, what would the test class look like?

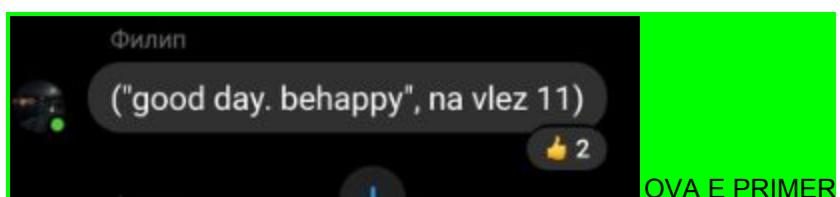
```
public class TestClass {
    @Test
    void getNumberTypeTest() {
        assertEquals("zero"
    }
}
```

(бидејќи е string затоа)

**ONA SHTO E
VNESENOST:**

CFG.getNumberType(3)
 CFG.getNumberType(-3)
 CFG.getNumberType(0)

TUKA STAVAME ONA SHTO OCHEKUVAME



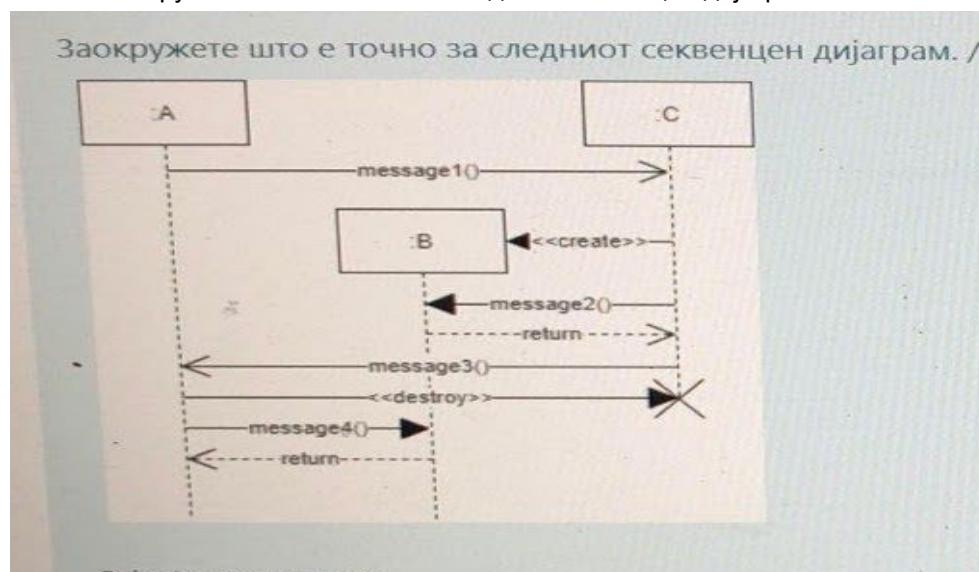
ОВА Е ПРИМЕР

Format

Ama namesto tekst treba broevi

ODGOVOR.

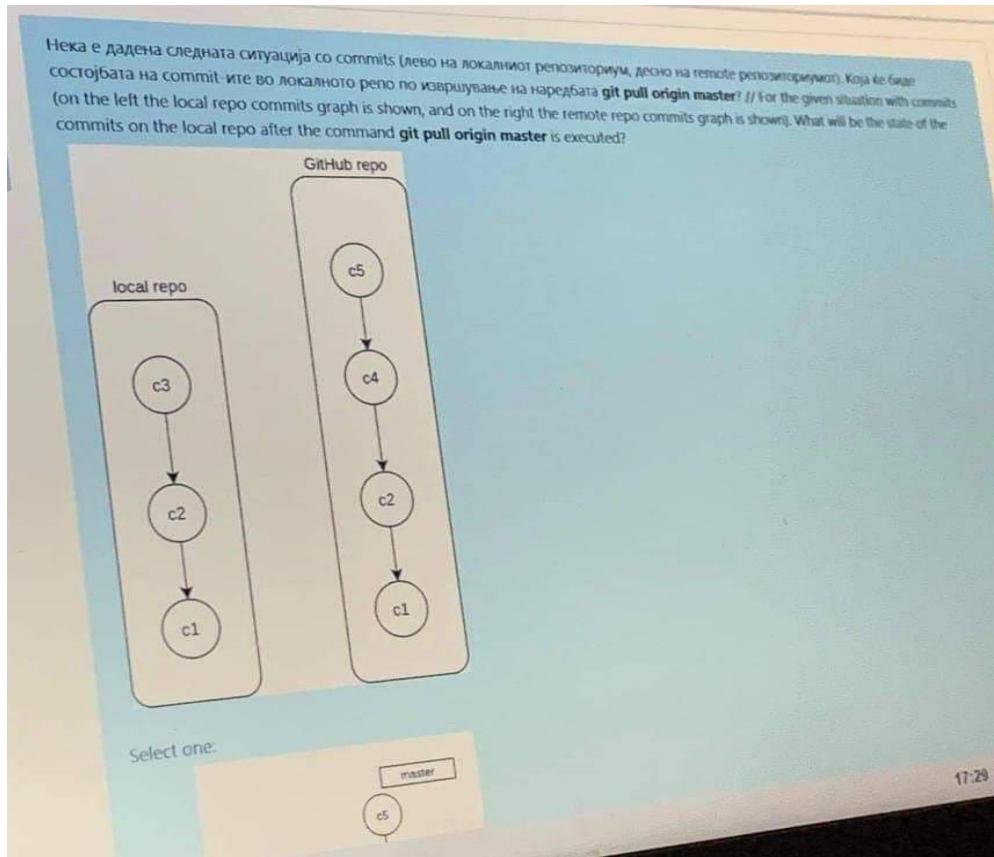
13. Заокружете што е точно за следниот секвенцен дијаграм.



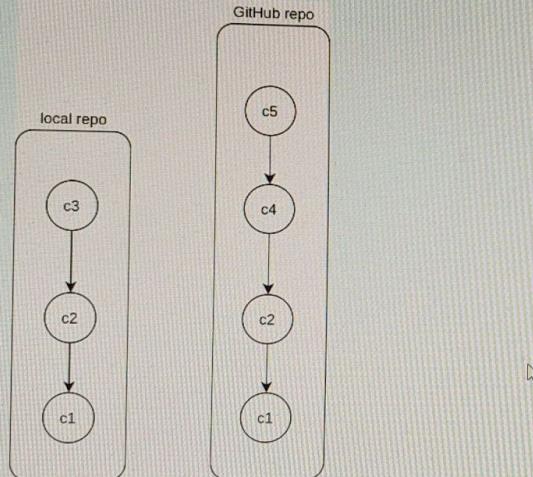
(слика за 13то прашанje)

- a. Пораката message3() е испратена по добивање повратна порака од пораката message2(). - ова е **точно**
- b.Пораката message2() е испратена по добивање повратна порака од пораката message1().
- c.**Објектот В не постои на почетокот на процесот кој е прикажан со овој дијаграм. - ова е точно**
односно е креiran
- d.Објектот С постои во моментот на испраќање повратна порака на пораката message4().
- e.Објектот В постои во текот на целиот процес кој е прикажан со овој дијаграм.

14. Нека е дадена следната ситуација со commits (лево на локалниот репозиториум, десно на remote репозиториумот). Која ќе биде состојбата на commit-ите во локално репо по извршување на наредбата **git pull origin master**?

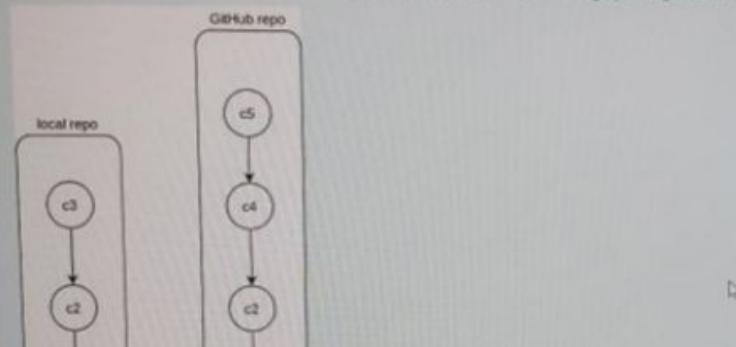


Нека е дадена следната ситуација со commits (лево на локалниот репозиториум, десно на ремоте репозиториумот). Која ќе биде состојбата на commit-ите во локалното репо по извршување на наредбата **git pull origin master?** // For the given situation with commits (on the left the local repo commits graph is shown, and on the right the remote repo commits graph is shown), What will be the state of the commits on the local repo after the command **git pull origin master** is executed?

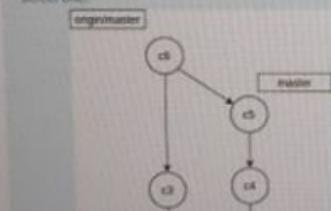


Select one:

Нека е дадена следната ситуација со commits (лево на локалниот репозиториум, десно на ремоте репозиториумот). Која ќе биде состојбата на commit-ите во локалното репо по извршување на наредбата **git pull origin master?** // For the given situation with commits (on the left the local repo commits graph is shown, and on the right the remote repo commits graph is shown), What will be the state of the commits on the local repo after the command **git pull origin master** is executed?

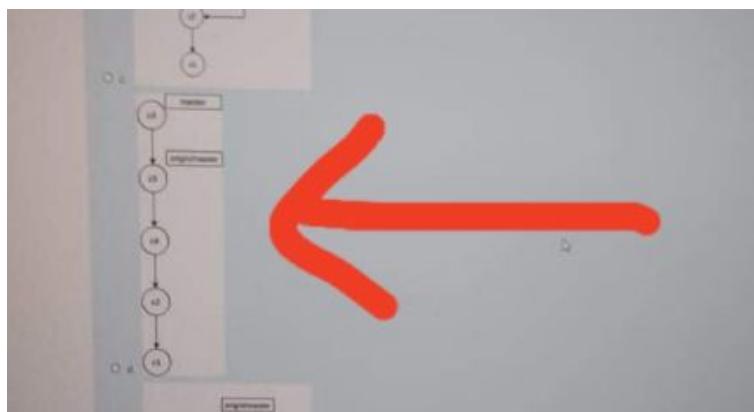


Select one:



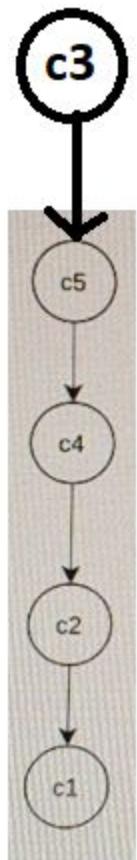
(slika za 14-to prashanje)

Одговор:



(slika za 14to prashanje)

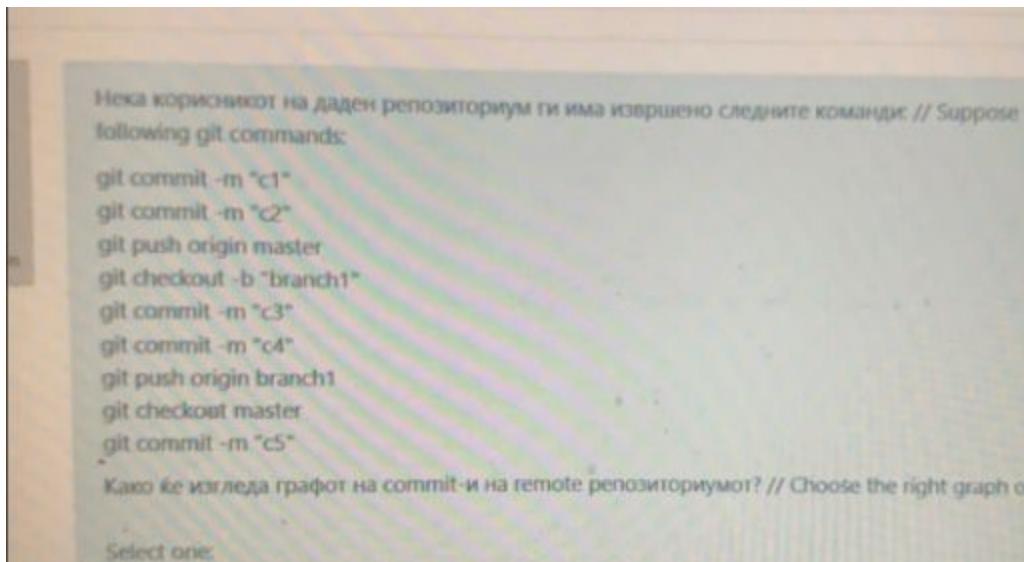
ODGOVOR



15. Нека корисникот на даден репозиториум ги има извршено следните команди:

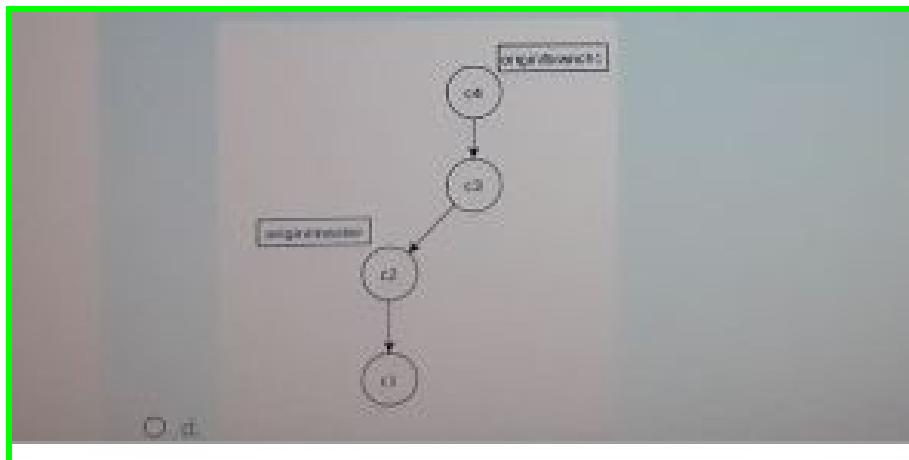
Git commit ... (на сликата)

Како ќе изгледа графот на commit-и на remote репозиториумот?



(slika za 15-to prashanje)

Одговор:

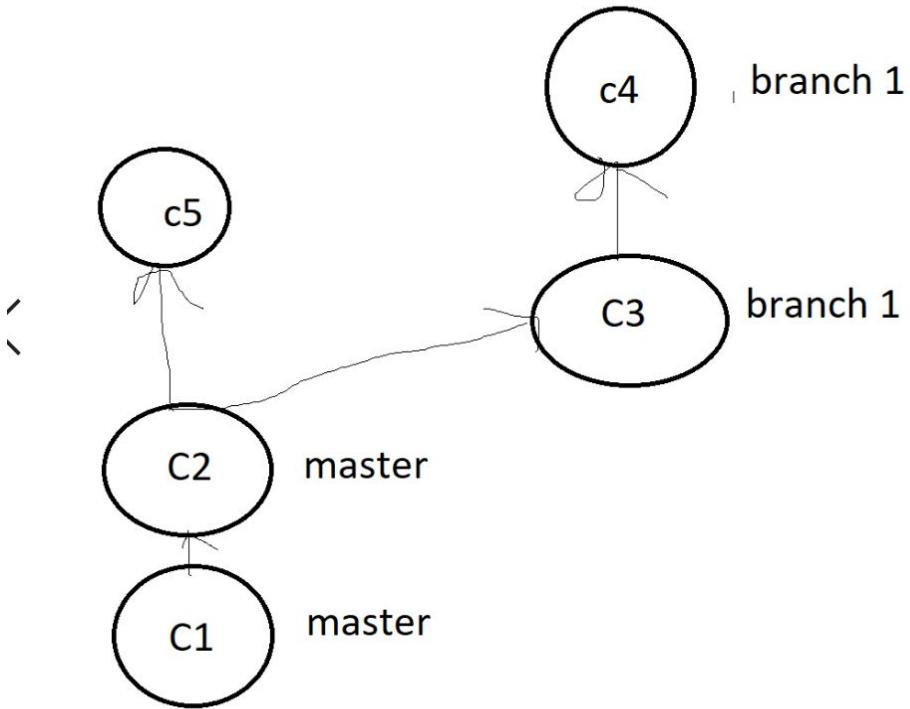


Tochno e pod d.

Mislam pod d zatoa sto posle c5 komitot ne e pushnat na remote

Dokolku beshe pushten, dokolku imashe push posle c5... kje beshe vaka?

-m e message, poraka





filip dev
2:23 PM Today



Prasanjeto e kako ke izgleda remote repo, taka da otkako e napraven commit na c5 na lokalniot master ne e promenata pratena so git push origin master. Znaci odgovorot si e pod D



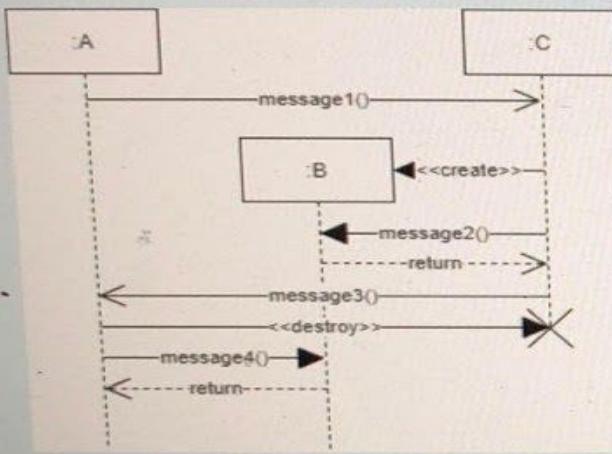
Viktor Kuzmanov
5:21 PM Today

tocno tako

Reply or add others with @

16. Заокружете што е точно од следниот секвенцен дијаграм?

Заокружете што е точно за следниот секвенчен дијаграм. /

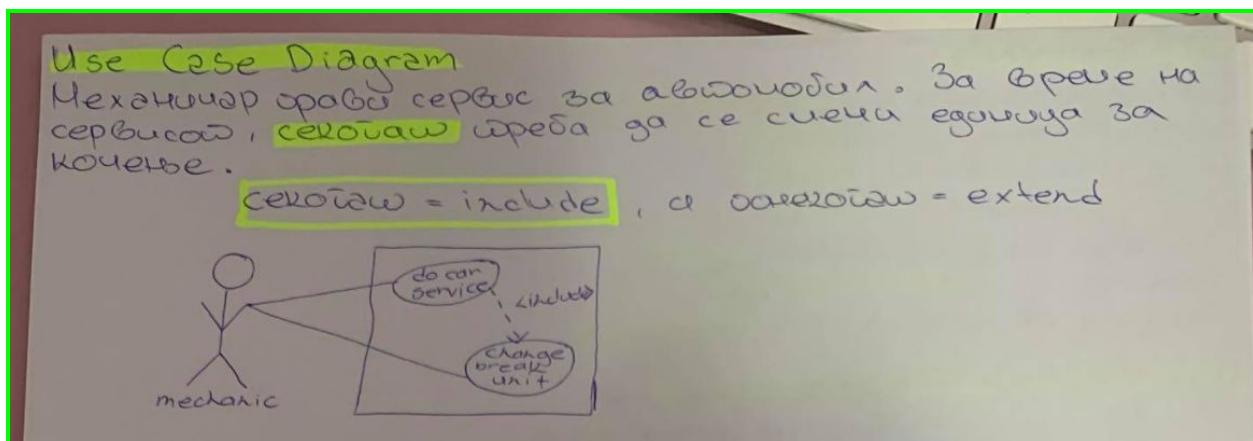


(slika za 16to prashanje)

- a. Објектот С постои во моментот на испраќање на повратната порака на пораката message4()
- b. Објектот А постои во текот на целиот процес кој е прикажан со овој дијаграм. - tochno
- c. Пораката message2() е испратена по добивање повратна порака од пораката message1().
- d. Објектот С не постои во моментот кога е повиката пораката message4() - tochno
- e. Објектот В постои во текот на целиот процес кој е прикажан со овој дијаграм.

+: Povratna poraka e so isprekinati linii

17. Како ќе го моделирате следното сценарио користејќи UML Use Case дијаграм? Механичар прави сервис за автомобил. За време на сервисот, **секогаш** треба да се смени и единицата за кочење.// **ако е sekogash e include, ако е ponekogash/mozhebi e extend...**



(od koleshka)

18. На slikata se pishani i dvete zadachi so git

Za prvata задача од slikata, за спојувањето на branch1 со master треба да пишува дека **nema da ima konflikt**

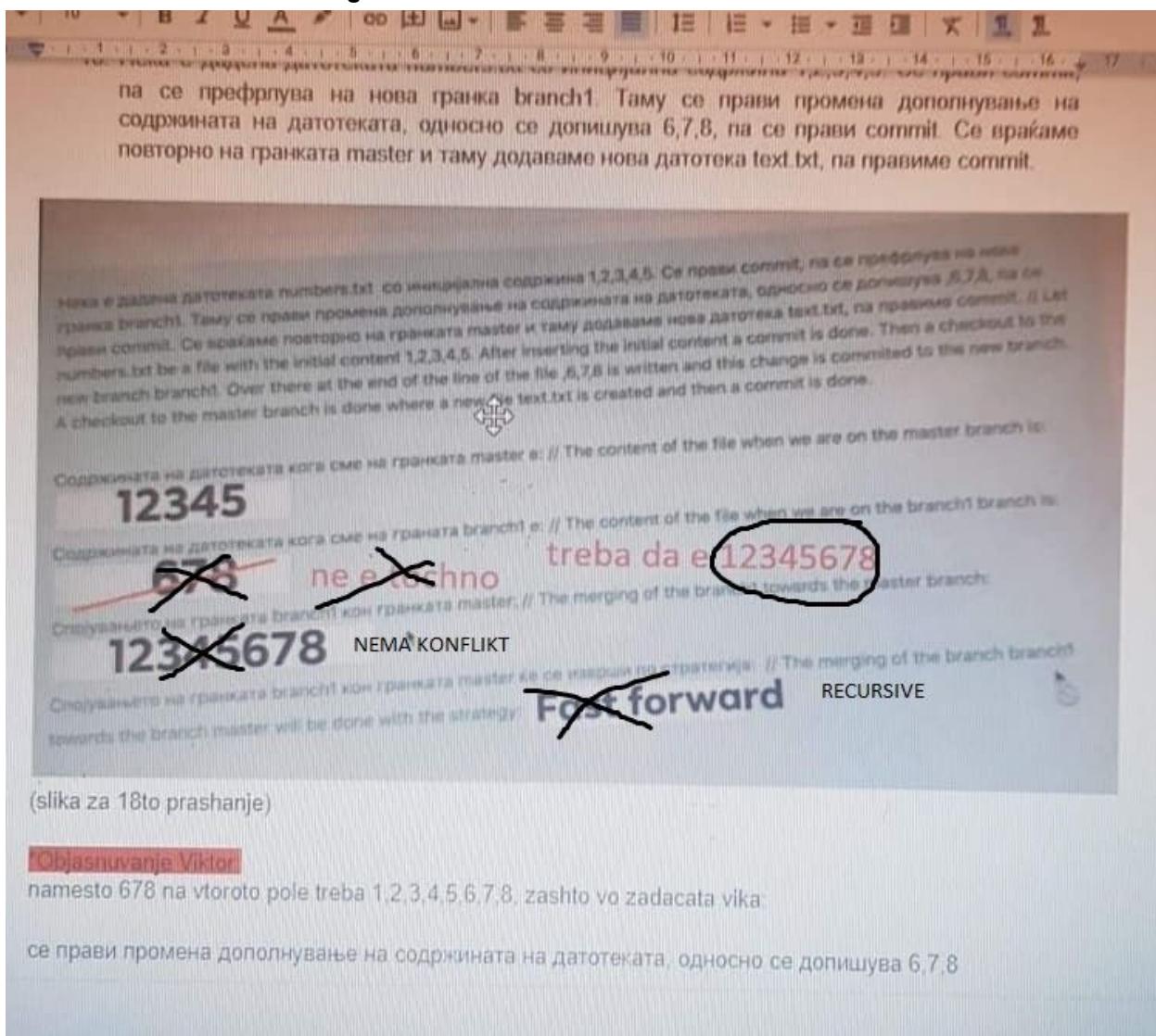
Za втората задача од slikata за спојувањето на branch1 со master исто така треба да пишува дека **nema da ima konflikt**, и дека стратегијата која се користи е **recursive merging - појасно е дадено со slikata подоле**

*Objasnuvanje Viktor:

namesto 678 на второто pole треба 1,2,3,4,5,6,7,8, zashto vo zadacata vika:

се прави промена дополнување на содржината на датотеката, односно се допишува 6,7,8 а не се prebrisuva strata sodrzina (1,2,3,4,5,.)

18. Ova e vtorata zadacha so git



Objasnuvanje: spojuvanjeto na grankata branch1 so master ke bide so rekurzivna strategija bidejki vo dvata brancha istovremeno se kreiralo nesto (na branch1 se men sodrzinata dodeka na master se dodal nov commit - file-ot text.txt) - Katerina

*REKURZIVNO E BIDEJKI ODI OD MASTER NA BRANCH1 PA NA MASTER PAK SE VRAKJA!

- Kako ponudeni opcii za spojuvanje na branch1 so master i vo dvete zadachi pogore se:

1. Ke ima konflikt
2. Ke nema konflikt

- Kako ponudeni opcii za koja strategija ke se koristi od dvete zadachi pogore se:

1. Fast-forwarding
2. Recursive merging
3. Ke javi konflikt

*Jane objasnuvanje:

nema posho ne cepka vo isti fajl

t.e ne modifcira isti fajl

posle prviot commit za broevite 1 2 3 4 5, kreira branch, istorijata e ista, tamu pravi promena vo toj fajl, si pravi commit, se vrakja i kreira nov fajl, si pravi commit

ako go modifciranje istiot fajl, ke imase konflikt, od 1 2 3 4 5 go smenil na 1 2 3 4 5 6, i koga merge, vo drugiot branch od 1 2 3 4 5 go smenil na 1 2 3 4 5 6 7 8

i ne znae koja promena da ja zacuva

inicijalno i dvete se bile 1 2 3 4 5 i 2 razlicni promeni ima napraveno

+

a bidejkji se vrakja pak nazad i kreira nova datoteka vo master?

tuka sepak nema konflikt?

nema da ima, ne e modifciran istiot fajl

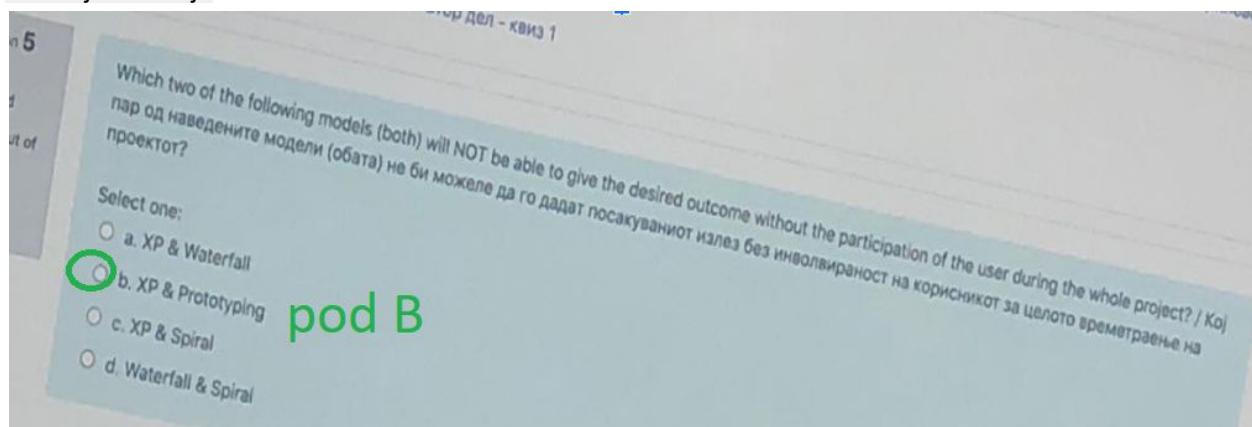
!Kako ne e modifciran istiot koga stanuva zbor za istiot fajl?

19. Koj par od наведените модели(обата) не би можеле да го дадат посакуваниот излез без инволвираност за целото времетраење на проектот?

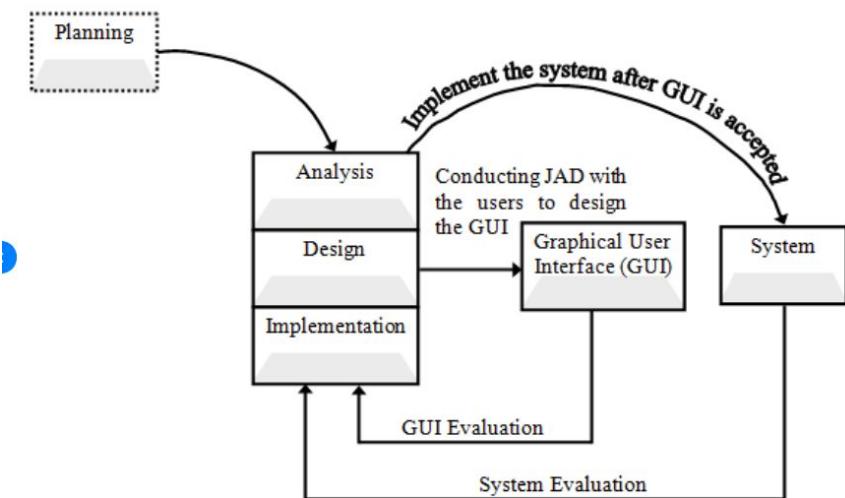
- a) XP & Waterfall - ne e tochno
- b) XP & Prototyping - ova e tochno
- c) XP & Spiral - ne e tochno
- d) Waterfall & Spiral - nee tochno

"Active Participation of user is involved in all the four phases of RAD model and in case of the Prototyping model we need user's presence/involvement every time a new prototype is build or designed"

Active Participation of user is involved in all the four phases of RAD model and in case of the Prototyping model we need user's presence/involvement every time a new prototype is build or designed.
eve objasnuvanje



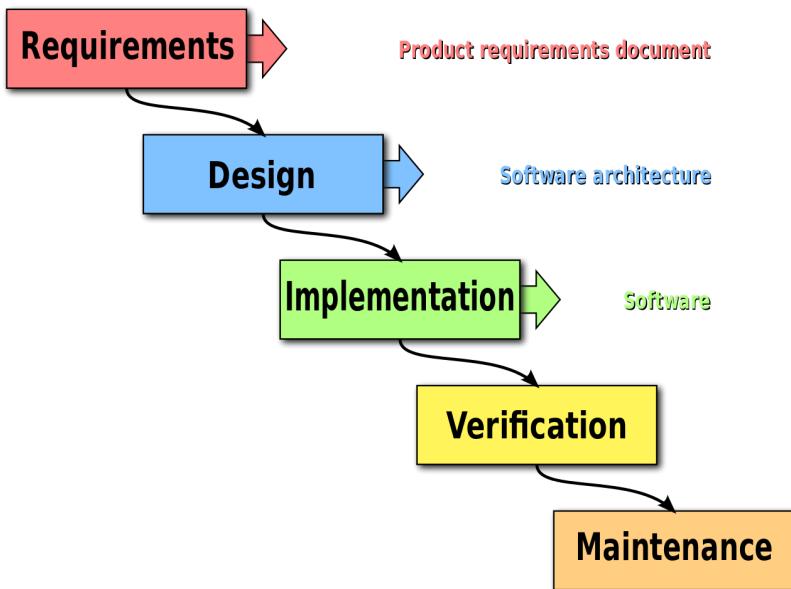
(слика за 19то прашање)



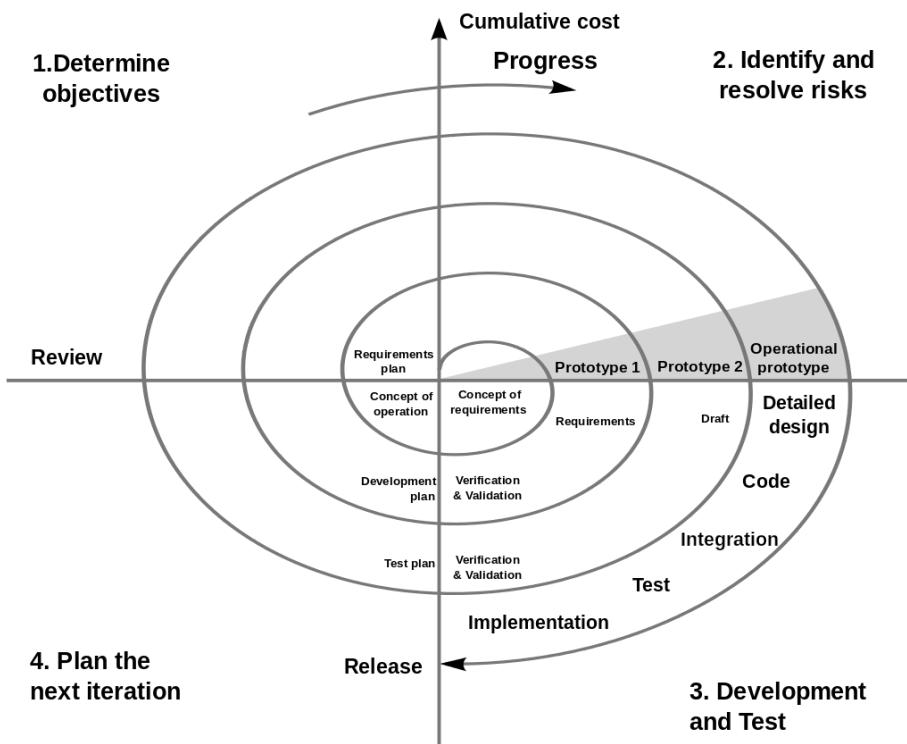
The Hybrid Method of " XP and Throwaway Prototyping "

*Waterfall i Spiral nemora da ima involviranjost

Waterfall:



Spiral:



20. Количеството неуспешност е клучно за софтверската надежност. Каква е неговата вредност додека софтверскиот производ е во употреба?

Failure rate is crucial for software reliability. How does it behave during the useful life of one software product?

- a) Се стабилизира секогаш кога го достигнува периодот на одвишност - ова е точно / it becomes stable whenever it reaches its time of obsolescence.
- b) Го достигнува својот минимум пред испораката - ова е точно / It reaches its minimum prior to delivery
- c) Повисока е отколку во периодот на тестирањето и дебагирањето
- d) Се зголемува по секоја надградба - ова е точно / it increases after each upgrade
- e) Се намалува по секоја надградба

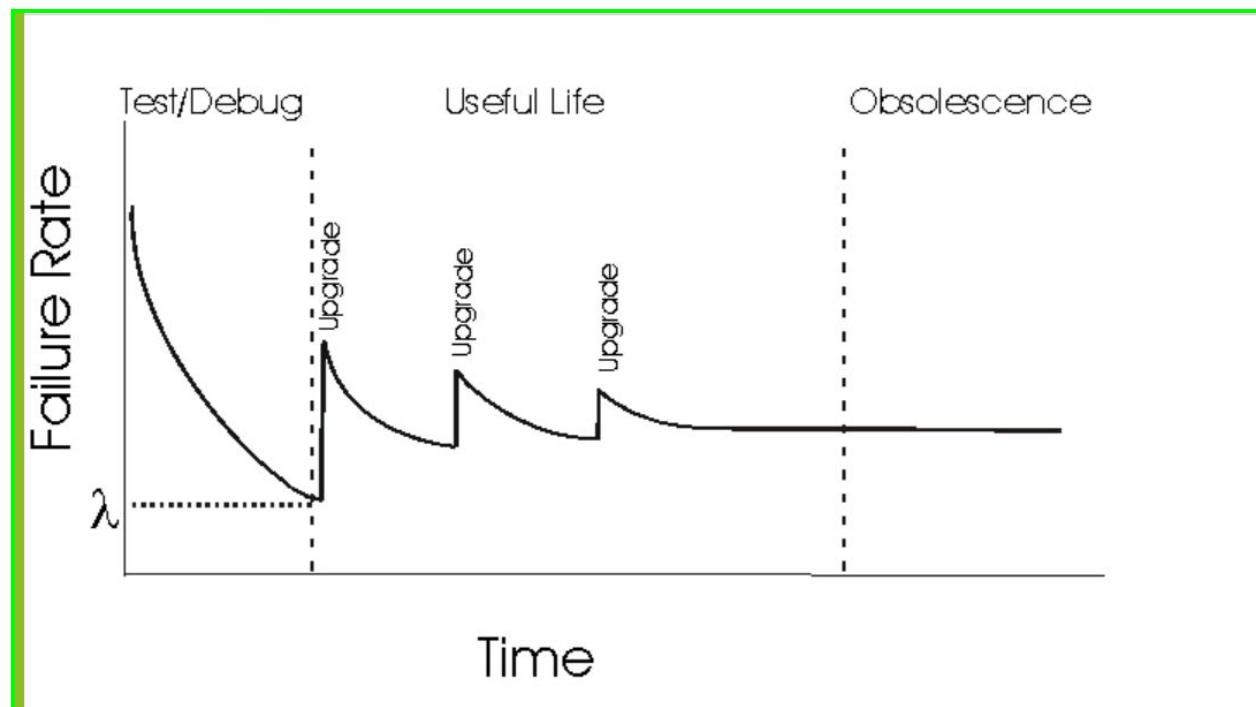
Failure rate is crucial for software reliability. How does it behave during the useful life of one software product? // Количеството неуспешност е клучно за софтверската надежност. Каква е неговата вредност додека софтверскиот производ е во употреба?

Select one or more:

- a. It becomes stable whenever it reaches its time of obsolescence // Се стабилизира секогаш кога го достигнува периодот на одвишност
- b. It reaches its minimum prior to delivery // Го достигнува својот минимум пред испораката
- c. It is higher than the failure rate during testing and debugging // Повисока е отколку во периодот на тестирањето и дебагирањето
- d. It increases after each upgrade // Се зголемува по секоја надградба
- e. It decreases after each upgrade // Се намалува по секоја надградба

(слика за 20то прашање)

Toчни се a,b i d.



definicija za obsolete

cause (a product or idea) to become obsolete by replacing it with something new.

pod b ima logika okej

fakticki nesto da e out of date/ne se koristi veke

i se zamenuva so nesto novo

so bi se koristelo

21. Кои се клучните проблеми кога при еволуцијата се применуваат агилните методи?

Точни се под c i d..

c.) Development and evolution team have different preferences about using agile methodology /

Тимовите за развој и еволуција даваат различна предност на агилните методологии.

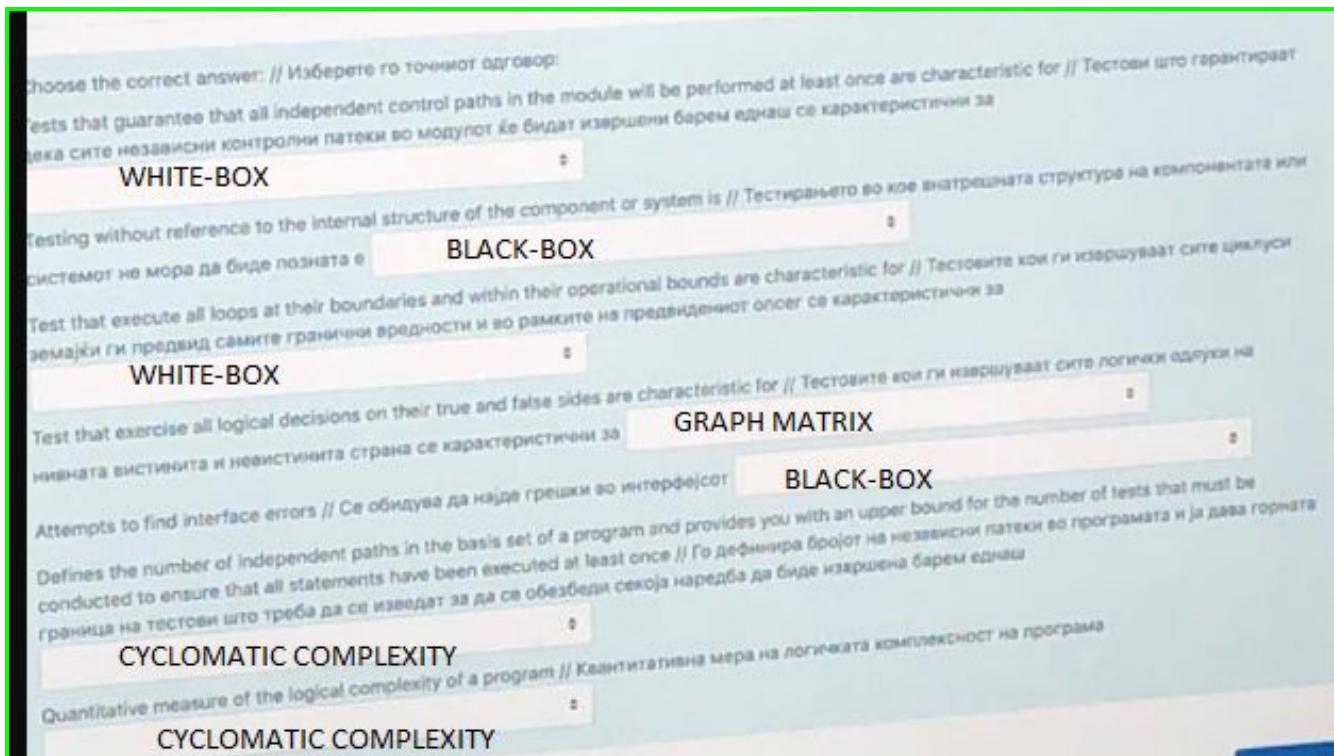
d.) Development and evolution team have different experience with agile methodology /

Тимовите за развој и еволуција имаат различно искуство со агилните методологии.

22. Што од следново НЕ се однесува на агилноста на софтверскиот процес?

Точно е под d. - Eliminate the use of project planning and testing / Се елиминира примената на планирање на проектот и тестирање.

23. Choose the correct answer: // Изберете го точниот одговор



(слика за 23то прашање)

*Тестови што гарантираат дека сите независни контролни патеки во модулот ќе бидат извршени барем еднаш се карактеристични за: white box

*Тестови во кое внатрешната структура на компонентата или системот не мора да биде позната е: black box

*Тестови кои ги извршуваат сите циклуси земајќи ги во предвид самите гранични вредности и во рамките на предвидениот опсег се: white box

*Тестови кои ги извршуваат сите логички одлуки на нивната вистинитосна и невистинита страна се: graph matrix

*Се обидува да се најде грешка во интерфејсот: black box

*HELP:

What is Black Box testing?

In Black-box testing, a tester doesn't have any information about the internal working of the software system. Black box testing is a high level of testing that focuses on the behavior of the software. It involves testing from an external or end-user perspective. Black box testing can be applied to virtually every level of software testing: unit, integration, system, and acceptance.

What is White Box testing?

White-box testing is a testing technique which checks the internal functioning of the system. In this method, testing is based on coverage of code statements, branches, paths or conditions. White-Box testing is considered as low-level testing. It is also called glass box, transparent box, clear box or code base testing. The white-box Testing method assumes that the path of the logic in a unit or program is known.

KEY DIFFERENCE

In Black Box, testing is done without the knowledge of the internal structure of program or application whereas in White Box, testing is done with knowledge of the internal structure of program.

Black Box test doesn't require programming knowledge whereas the White Box test requires programming knowledge.

Black Box testing has the main goal to test the behavior of the software whereas White Box testing has the main goal to test the internal operation of the system.

Black Box testing is focused on external or end-user perspective whereas White Box testing is focused on code structure, conditions, paths and branches.

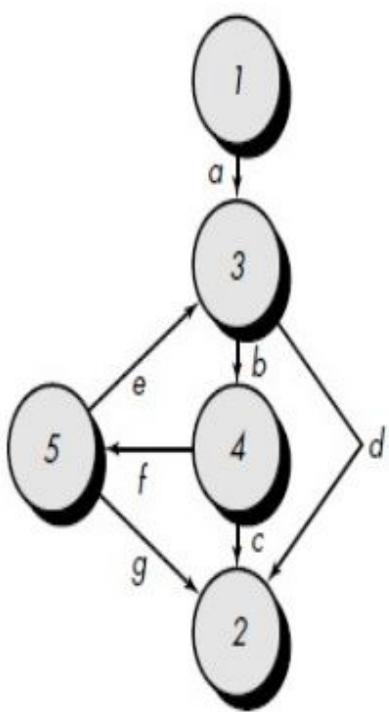
Black Box test provides low granularity reports whereas the White Box test provides high granularity reports.

Black Box testing is a not time-consuming process whereas White Box testing is a time-consuming process.

GRAPH MATRIX:

GRAPH MATRICES

- To develop a software tool that assists in basis path testing, a data structure, called a *graph matrix*, can be quite useful.
- A *graph matrix* is a square matrix whose size (i.e., number of rows and columns) is equal to the number of nodes on the flow graph. Each row and column corresponds to an identified node, and matrix entries correspond to connections (an edge) between nodes.
- The graph matrix is nothing more than a tabular representation of a flow graph.



Flow graph

Node	1	2	3	4	5
1			a		
2					
3		d		b	
4		c			f
5	g	e			

Graph matrix

+cyclomatic complexity:

Cyclomatic Complexity

- The number of tests to test all control statements equals the **cyclomatic complexity**
- **Cyclomatic complexity** equals number of conditions in a program plus one (or equivalently, in the program flow graph it is the “Number of edges - Number of nodes +2”)
- **Conditions** are any type of branching operation such as each “if” statement or any types of loop (for, while etc.)
- Useful if used with care. Does not imply adequacy of testing. Although all paths are executed, all combinations of paths are not executed



Edges and Nodes Method:

$$v = e - n + 2$$

$$e = 12, n = 8$$

$$v = 12 - 8 + 2 = 6$$

Predicate Method:

$$v = \sum \pi + 1$$

$\sum \pi = 5$, sum of predicates

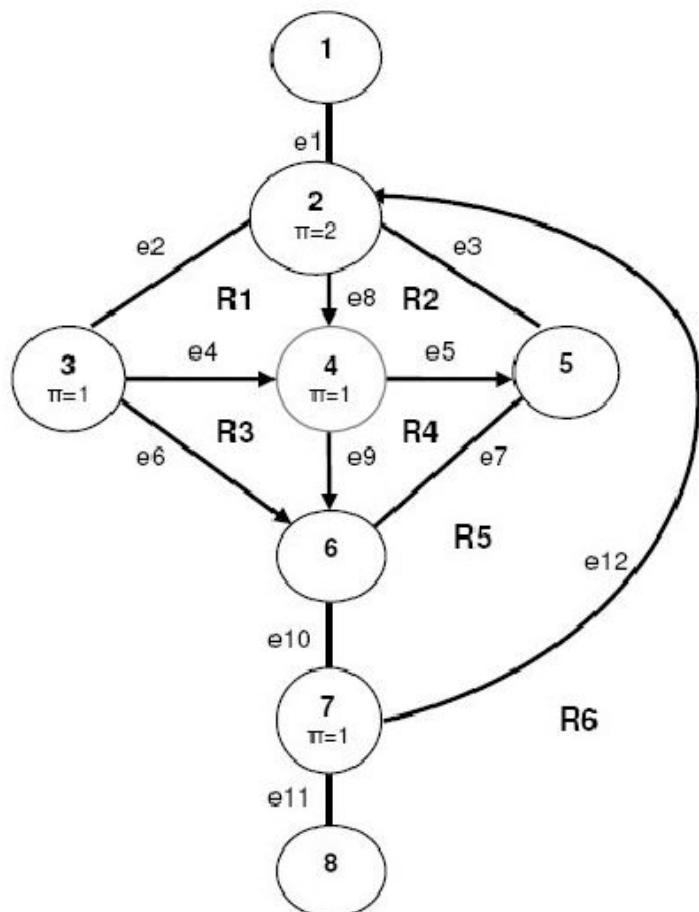
$$v = 5 + 1 = 6$$

Region (Topological) Method:

$$v = \sum R, \text{ sum of regions } R$$

$$\sum R = 6$$

$$v = 6$$



24. Што претставува програмирање во пар? / What is Pair Programming?

Question 17

Not yet
answered

Marked out of
2.00

1' Flag
question

What is Pair Programming? / Што претставува програмирање во пар?

Select one:

- a. A development method, where two developers work on the same workstation. / Процес на развој на софтвер во кој два развивачи на софтвер работат на иста работна станица.
- b. It is a programming process, wherein a programmer writes the program twice just in case. / Процес на програмирање во кој програмерот ја пишува програмата два пати, за секој случај.
- c. It is a way of rectifying errors in programs by looking for pairs of errors. / Начин за поправување на грешки во програма барајќи парови на грешки.
- d. It is an algorithm that finds pair numbers in a set of integers. / Алгоритам кој пронаоѓа парови на броеви во множество на цели броеви.

Next page

A

Previous page

Соби 1 ►

(Slika za 24to prashanje)

Tochno e pod a. - A development method, where two developers work on the same workstation / Процес на развој на софтвер во кој два развивачи на софтвер работат на иста работна станица.

25. Fali prashanje

26. Квалификацијата на компоненти гарантира:

Component qualification ensures: // Квалификацијата на компоненти гарантира

Select one:

b. that a candidate component will fulfill all performance criteria required for the application // дека компонента-кандидат ќе ги исполни сите потребни критериуми за извршување на функциите на апликацијата

c. that a candidate component will perform the function required for the application // дека компонента-кандидат ќе ги изврши функциите коишто се потребни за апликацијата

d. that a candidate component will fulfill all design criteria required for the application // дека компонента-кандидат ќе ги исполни сите потребни критериуми за комуникација со надворешен систем

(Slika za 26to prashanje)

Tochno e pod б. - That a candidate component will perform the function requirement for the application /
дека компонента-кандидат ќе ги изврши функциите коишто се потребни за апликацијата.

27. Select the correct characteristics of the software regarding its testability? // Изберете го точниот атрибут на софтверот од аспект на можноста за негово тестирање?

Select the correct characteristic of the software regarding its testability? // Изберете го точниот атрибут на софтверот од аспект на можностите за негово тестирање?

Relatively few bugs will block the execution of tests. // Нема грешки кои го блокираат извршувањето на тестови.

Operability // Операбилност

Internal errors are automatically detected and reported. // Интерните грешки автоматски се фалат и пријавуваат.

OBSERVABILITY

All possible outputs can be generated through some combination of input, and I/O formats are consistent and structured. // Сите можни излози може да бидат генерираны преку некоја комбинација на влезови, а влезно/излозните формати се конзистентни и структурирани.

CONTROLABILITY

The software system is built from independent modules that can be tested independently. // Софтверскиот систем е изграден од независни модули што може да се тестираат независно.

DECOMPOSABILITY

Architecture is modularized to limit the propagation of faults. // Архитектурата е модуларизирана за да се намали пропагацијата на грешки.

SIMPLICITY

The software recovers well from failures. // Софтверот добро се спираува со испади.

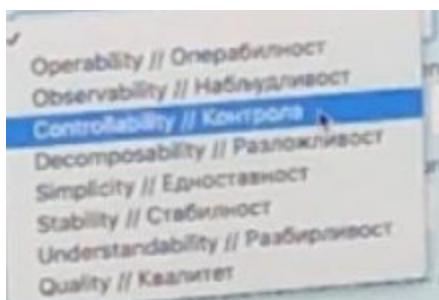
STABILITY

Technical documentation is instantly accessible, well organized, specific and detailed, and accurate. // Техничката документација е лесно достапна, добро организирана, детална, прецизна и точна.

UNDERSTANDABILITY

Next page

(Slika za 27мо прашанje)



28. Кои од следните не е проблем при дизајн на кориснички интерфејс?

Which one is not an user interface design issue? // Кои од следните не е проблем при дизајн на кориснички интерфејс?

Select one:

- a. Response time // време на одзив
- b. Error handling // справување со грешки
- c. Internationalization // интернационализација
- d. Memory usage // мемориска искористеност

D

(Slika za 28mo prashanje)

Tochno e pod d. - Memory usage / мемориска искористеност

29. Што од следново е точно за слоевита архитектура?

Which of the following is true with respect to layered architecture? / Што од следново е точно за слоевита архитектура?

Select one:

- a. A layer may call other layers above and below it, as long as it uses them / Даден слој може да ги повикува слоевите над и под него се додека ги користи
- b. Each layer is allowed to use all the layers "above" it / Секој слој може да ги користи сите слоеви „над“ него
- c. Each layer is allowed to depend on the layer above it being present and correct / Дозволено е секој слој да зависи од слојот над него ако истиот е присутен и коретен
- d. Each layer provides services to the layer "above" and makes use of services provided by the layer "below" / Секој слој обезбедува услуги на слојот „над“ него и ги користи услугите на сервисите обезбедени од слојот „под“ него

D

(Slika za 29to prashanje)

Tochno e pod d - Each layer provides services to the layer "above" and makes use of services provided by the layer "below". / Секој слој обезбедува услуги на слојот „над“, него и ги користи услугите на сервисите обезбедени од слојот „под“, него.

30. Кој поглед во архитектонскиот дизајн ги прикажува клучните апстракции во системот како објекти или класи од објекти?

СТВО-2019/2020/L

6 / Втор дел од јунскиот испит, 14.07.2020 / Втор дел - квиз 1

Which view in architectural design shows the key abstractions in the system as objects or object classes? / Кој поглед во архитектонскиот дизајн ги прикажува клучните апстракции во системот ако објекти или класи од објекти?

Select one:

- a. physical / физички
- b. development / развоен
- c. process / процесен
- d. logical / логички

[Next page](#)

(Slika za 30то прашанje)

Tochno e pod d - Logical/ логички.

31. Кои од следните изрази е точен за задачите во Gradle?

-2019/2020/L

Кои од следните изрази е точен за задачите во Gradle? // Which of the following statements is correct for the Gradle tasks.

Select one or more:

- a. Од задачите и нивните зависности може да се креира директен ацикличен граф, каде што тасковите се јазлите а зависностите се ребрата на графот. // From the tasks and their mutual dependencies we can create a Directed Acyclic Graph, where the nodes are the tasks and the edges are the dependencies between them.
- b. Сите задачи мора да се независни // All tasks must be independent
- c. Агрегат задачи се задачи коишто не превземаат никакви акции туку само зависат од други задачи. // Aggregate tasks are tasks who don't take any actions, but they only depend on other tasks.
- d. Доколку една задача не се изврши успешно, не се извршуваат успешно и сите задачи од коишто таа задача зависи. // If a task fails then all the tasks on which the main task depends, fail as well.
- e. Секоја задача мора да биде зависна од некоја друга задача // Every task has to depend on some other task
- f. Секоја задача мора да има и акции и влез и излез. // Every task has to have actions, input and output.
- g. Задачите коишто се посветени на делови од стандардниот животен циклус на софтверот не зависат од други задачи // The tasks dedicated to standard software life cycle segments, don't depend on other tasks.

Next page

(Slika za 31vo prashanje)

Tochno e pod a,c i f..

32. Кои од активностите се важни за инженерство на доменот:

Which activities are important for domain engineering // Кои од активностите се важни за инженерство на доменот

Select one or more:

- a. Analysis // анализа
- b. Construction // конструкција
- c. Testing // тестирање
- d. Implementation // имплементација

(Slika za 32ro prashanje)

Tochno e pod a, b i d. - Анализа, конструкција и имплементација.

33. Моделирање на однесување треба да претставува:

Behavioral modeling should present // Моделирање на однесување треба да претставува:

Select one:

- a. how the system responds to internal events // како системот реагира на внатрешни настани
- b. how the system responds to external events // како системот реагира на надворешни настани
- c. how the system responds to testing // како системот реагира на тестирање
- d. how the system responds to user interactions // како системот реагира на интеракција со корисникот

(Slika za 33to prashanje)

Tochno e pod b. - Кајо системот реагира на надворешни настани.

34. Колаборационите дијаграми се дел од:

Question 20
Not yet answered
Marked out of 2.00
Flag question

Collaboration diagrams are part of the // Колаборационите дијаграми се дел од

Select one:

- a. Behavioral models // Модели на однесување
- b. Class models // класните модели
- c. Flow models
- d. Scenario-based models // моделите базирани на сценарио

(Slika za 34to prashanje)

Tochno e pod b. - Класните модели.

35. Анализата на интерфејсот треба да ни даде информации за:

Question 2
Not yet answered
Marked out of 2.00
Flag question

Interface analysis should give us information about // Анализата на интерфејсот треба да ни даде информации за:

Select one or more:

- a. the people who will use the system // луѓето што ќе го користат системот
- b. the tasks of the users that they will perform on the system // задачите коишто корисниците ќе ги изведуваат на системот
- c. the front end library for presenting the informations // библиотеката што ќе се користи за претставување на информациите на front end
- d. the complexity of the implementation of the interface // комплексноста на имплементацијата на интерфејсот

(Slika za 35to prashanje)

a)the people who will use the system//Луѓето што ќе го користат системот.

b)the tasks of the users that they will perform on the system/Задачите коишто корисниците ќе ги изведуваат на системот.

36. Како ќе го моделирате следното сценарио користејќи UML Use Case дијаграм?

Како ќе го моделирате следното сценарио користејќи UML Use Case дијаграм?

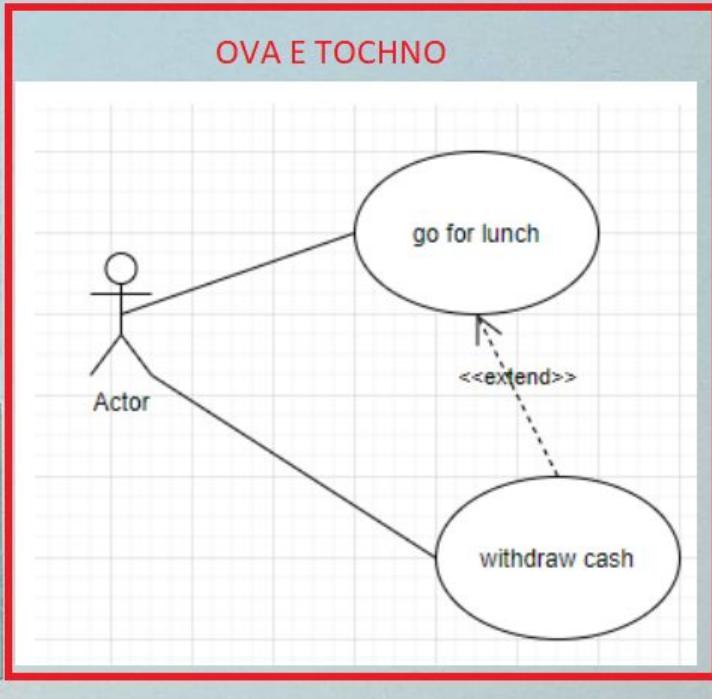
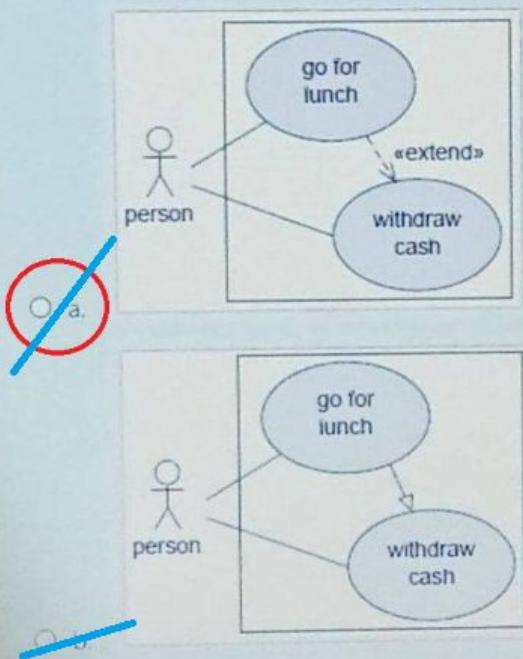
Едно лице оди на ручек. Поради тоа, на лицето може да му биде потребно да повлече пари од банкоматот.

/

How do you model the following situation using UML Use Case diagram?

A person goes for lunch. In the course of that it might be necessary that the person withdraws cash from an ATM.

Select one:



(Slika za 36to prashanje/ne e tochno)

Odgovor 36to=ova e tochno

37. Претпоставете дека, во рамки на некој git репозиториум, изгледот на stashing stack е како што е прикажан подолу.

Како ќе се промени изгледот на stashing stack (односно каков ќе биде излезот од повикот на командата git stash list) по извршување на следните команди:

Предпоставете дека, во рамки на некој git репозиториум, изгледот на stashing stack е како што е прикажано подолу. / Assume that in a git repository the stash stack looks as follows.

stash@{0}: WIP on master: 567befg Extract embeddings
stash@{1}: WIP on master: 4567def Read word pairs
stash@{2}: WIP on master: 3456cde Create vocabulary
stash@{3}: WIP on master: 2345bcd Add ReadMe
stash@{4}: WIP on master: 1234abc Initial commit

Како ќе се промени изгледот на stashing stack (односно каков ќе биде излезот од повикот на командата git stash list) по извршување на следните команди: / What will be the output of the command git stash list after executing the following commands:

git stash apply
git stash drop stash@{1}
git stash pop

Select one:

- a. stash@{0}: WIP on master: 567befg Extract embeddings
stash@{2}: WIP on master: 3456cde Create vocabulary
stash@{3}: WIP on master: 2345bcd Add ReadMe
stash@{4}: WIP on master: 1234abc Initial commit
- b. Нема да има промена на изгледот / There will be no changes to the stash stack.
- c. stash@{3}: WIP on master: 2345bcd Add ReadMe
stash@{4}: WIP on master: 1234abc Initial commit
- d. Командите не се валидни / The commands are invalid
- e. stash@{2}: WIP on master: 3456cde Create vocabulary
stash@{3}: WIP on master: 2345bcd Add ReadMe
stash@{4}: WIP on master: 1234abc Initial commit

ova e tochno, pod E

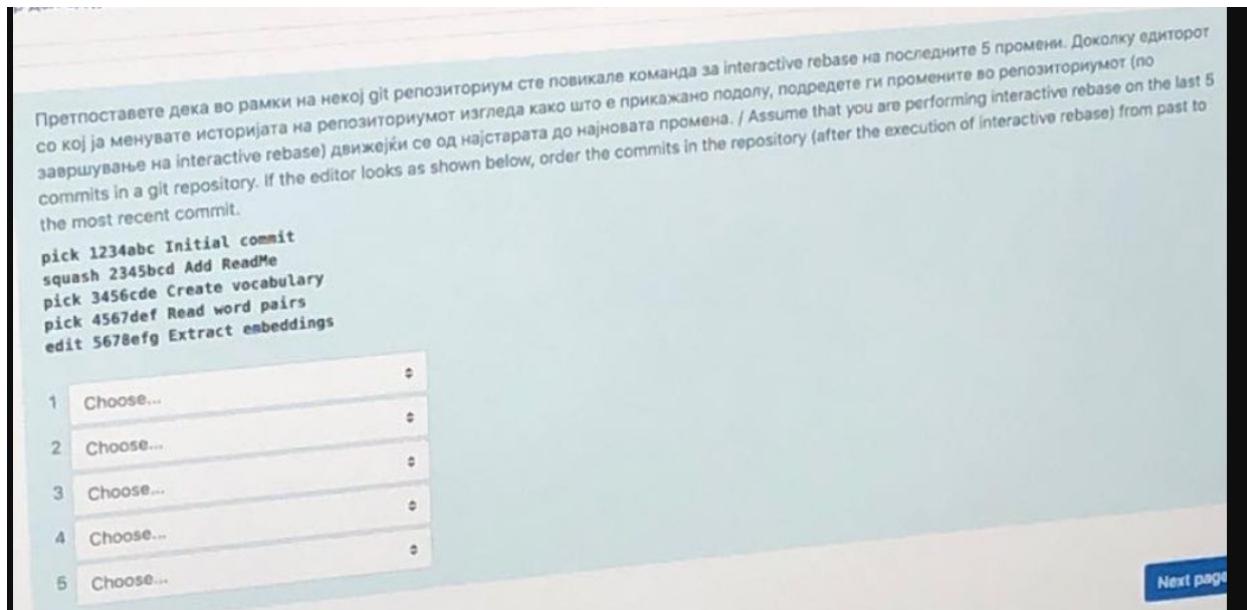
[Next page](#)

(slika na 37mo prashanje)

Tochno pod e.

git stash apply ги става промените од stash, но не ги трнува од stashot
така да посle тaa команда истi ke ostane stashot
so drop ke go izbrise prviot, i so pop ke go popne vtoriot
така да 2 ke se otstranat od stashot
prvo ke se dropne stash{1}, a so pop ke se dropne stash{0}

38. Претпоставете дека во рамки на некој git репозиториум сте повикале команда за interactive rebase на последните 5 промени. Доколку едиторот со кој ја менувате историјата на репозиториумот изгледа како што е прикажано подолу, подредете ги промените во репозиториумот (по извршување на interactive rebase) движејќи се од најстарата до најновата промена.



(slika za 38mo prashanje)

Kire - tochen odgovor

- 1. Vo prviot cekor kaj initial commit nema nisto da se sluci,
- 2. potoa kaj squash kje gi spoi vo eden commit (initial commit i add readme)
- 3. Create vocabulary nista nema da se sluci bidejkji ima pick napred
- 4. Read word pairs nista nema da se sluci bidejkji ima pick napred
- 5. So edit moze da se splitnat dva commits.

Pa spored mene bi imale

1. Initial commit i readme -spoeni so squash bidejkji squash go spojuva tekovniot commit so predhodniot
2. Create vocabulary
3. Read word pairs
4. Extract embeddings
5. Extract embeddings kaj poslednite dve ne znam sto bi splitnalo (ne znam sto ima od ponudenite odgovori) mozebi embedding1 i embedding2 bi bile 4 i 5 soodvetno.

Plus pisuva da se naredat od најстарата до најновата промена тоа би изгледало вака како што пишувате а доколку во git пишете git log --oneline каде ги нареди од долне па нагоре по старина на commitot т.e. во обратен редослед од напишаното.

GRETA - najnovo:

initial and add read me

create vocabulary

read word pairs

extract embeddings (posto napred ima edit mozno e commitot da e razdelen na dva

posebni, pa ako ima ponudeno extract kje bide 4to, a embedding kje bide 5to)

commitot kaj koj ima squash se spojuva so toj nad nego

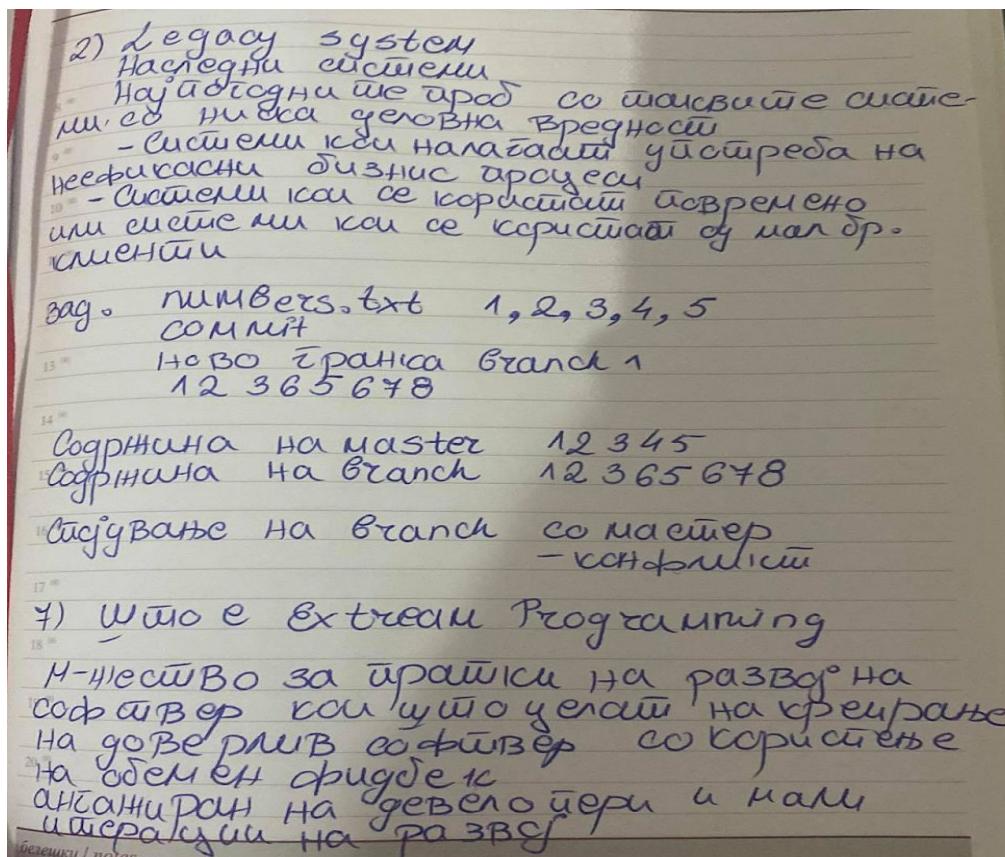
a za poslednoto extract embedding se zavisi sto ima na drop down...

ВТОР ДЕЛ:

<https://www.sanfoundry.com/software-engg-mcqs-software-engineering-ethics-1/>

+ Од колешка:

+



+ Од колега:

```
* f60248c c2
* aab3777 c1
* 83d484b first commit

Filip@Filip MINGW64 ~/Desktop/5est/5est2 (master|Merging)
$ git log --graph --oneline --decorate --all
* 8af5c6c (HEAD -> master) c3
| * e5003fa (origin/master) c5
| * 9c37a03 Create c4
|
* f60248c c2
* aab3777 c1
* 83d484b first commit

Filip@Filip MINGW64 ~/Desktop/5est/5est2 (master|Merging)
$ |
```

Umbrella Activities

Дополнителни константни активности кои треба да обезбедат успешно и непречено одвивање на процесот:

- ◆ Tracking and control - Следење и контрола на софтверскиот проект
- ◆ Risk management - Менаџирање на ризици
- ◆ SQA (Software quality assurance) – Активности за да се обезбеди квалитет на софтверот
- ◆ Technical reviews - Формални технички прегледи (ревизии)
- ◆ Measurement – дефинирање и колекција на процесни и продуктни мерки
- ◆ Configuration management (менажирање на ефектите на промена)
- ◆ Reusability management (критериуми за повторна употреба)
- ◆ Подготовка и продукција на работни материјали (модели, документи, записници, формулари, ...)

NOVI:

1. Кои се четирите клучни професионални обврски? - Стручност, доверливост, злоупотреба на компјутерите, право на интелектуална сопственост.

Кои се четирите клучни професионални обврски? / Which are the four major issues of professional responsibility?

Select one or more:

- a. Стручност, право на приватност, право на безбедност / Competence, Privacy rights, Security rights
- b. Стручност, доверливост, злоупотреба на компјутерите / Competence, Confidentiality, Computer misuse
- c. Стручност, злоупотреба на компјутерите, право на интелектуална сопственост / Competence, Computer misuse, IP rights
- d. Злоупотреба на компјутерите, право на интелектуална сопственост, право на приватност / Computer misuse, IP rights, Privacy rights

Кои се четирите клучни професионални обврски? / Which are the four major issues of professional responsibility?

Select one or more:

- a. Злоупотреба на компјутерите, право на интелектуална сопственост, право на приватност / Computer misuse, IP rights, Privacy rights
- b. Стручност, право на приватност, право на безбедност / Competence, Privacy rights, Security rights
- c. Стручност, доверливост, злоупотреба на компјутерите / Competence, Confidentiality, Computer misuse
- d. Стручност, злоупотреба на компјутерите, право на интелектуална сопственост / Competence, Computer misuse, IP rights

C, D

Next page

POINAKU SE PONUDENI TUKA

2. ISO стандардот ги гарантира квалитетот, безбедноста и ефикасноста на продуктите, сервисите и системите. Кои од наведените ИСО стандарди не се насочени само кон софтвер?**ISO 9000 family**

ISO стандардот ги гарантира квалитетот, безбедноста и ефикасноста на продуктите, сервисите и системите. Кои од наведените ИСО стандарди не се насочени само кон софтвер? (ISO standards ensure the quality, safety, and efficiency of products, services, and systems. Which of the following ISO standards are not only software directed?)

Select one or more:

- a. ISO/IEC 15504-5
- b. ISO 9000 family
- c. ISO/IEC 90003:2014
- d. ISO 25010

ZNACHI B I C se tochni

i. ISO/IEC 90003: 2014 – дава насоки на организациите за примената на ISO 9001:2008 за стекнување, снабдување, развој, работење и одржување на компјутерски софтвер и сродна поддршка на услуги.

-Примена на ISO/IEC 90003: 2014 соодветен за:

1. Дел од трговски договор со друга организација
2. Производ достапен за пазарниот сектор.
3. Се користи како поддршка на процесите во една организација
4. Вграден во хардверски производ
5. Поврзани со софтверски услуги.

created in response to customer or project requirements. For example, standards for formal specifications may be required if these standards have not been used in previous projects.

24.2.1 The ISO 9001 standards framework

The international set of standards used in the development of quality management systems in all industries is called ISO 9000. ISO 9000 standards can be applied to a range of organizations from manufacturing through to service industries. ISO 9001, the most general of these standards, applies to organizations that design, develop, and maintain products, including software. The ISO 9001 standard was originally developed in 1987. I explain the 2008 version of the standard here, but the standard may change in 2015 when a new version is scheduled for release.

The ISO 9001 standard is not a standard for software development but rather is a framework for developing software standards. It sets out general quality principles, describes quality processes in general, and lays out the organizational standards and procedures that should be defined. These should be documented in an organizational quality manual.

A major revision of the ISO 9001 standard in 2000 reoriented the standard around nine core processes (Figure 24.5). If an organization is to be ISO 9001 conformant, it must document how its processes relate to these core processes. It must also define and maintain records demonstrating that the defined organizational processes have

3. Која е кардиналноста на релацијата помеѓу ентитетите студент и професор (на еден факултет)?

Која е кардиналноста на релацијата помеѓу ентитетите студент и професор (на еден факултет)? / What is the cardinality of the relationship between the entities student and professor (on one faculty)?

Select one:

- a. 1:M
- b. M:1
- c. 1:1
- d. M:M

B **D**

- Еден колега мисли M:M - TOCHNO E M:M! Many-to-many! ZNACHI POD D.

4. Што означува кратенката CASE во софтверското инженерство?

- Софтверски системи наменети да обезбедат автоматизирана поддршка за софтверските процесни активности.

Што означува кратенката CASE во софтверското инженерство? / What does CASE mean in software engineering?

Select one:

- a. Copy And Steal Everything
- b. Софтверски системи наменети да обезбедат автоматизирана поддршка за софтверските процесни активности / Software systems intended to provide automated process activities
- c. Вообичаните елементи за апликациските сервиси / Common Application Service Elements
- d. Проценка на усогласеноста на сигурносните аспекти / Conformity Assessment Security Evaluation

B

5. Согласно со извештајот на Global Information Technology Industry, глобалната вредност на ИТ индустријата надминува 5 трилиони американски долари. Споредена со земјоделето, ИТ индустријата вреди:

Согласно со извештајот на Global Information Technology Industry, глобалната вредност на ИТ индустријата надминува 5 трилиони американски долари. Споредена со земјоделето, ИТ индустријата вреди: / According to Global Information Technology Industry, IT industry global value is over 5 trillion US\$. Compared with agriculture, IT industry worth is:

Select one:

- a. повеќе од двојно / more than twice as much
- b. приближно еднакво / approximately the same
- c. скоро двојно повеќе / almost twice as much
- d. половина од вредноста на земјоделието / half the value of agriculture

Clear my choice

A

POD A E TOCHNO zemjodelska e 2.4 trilioni a IT 5.2

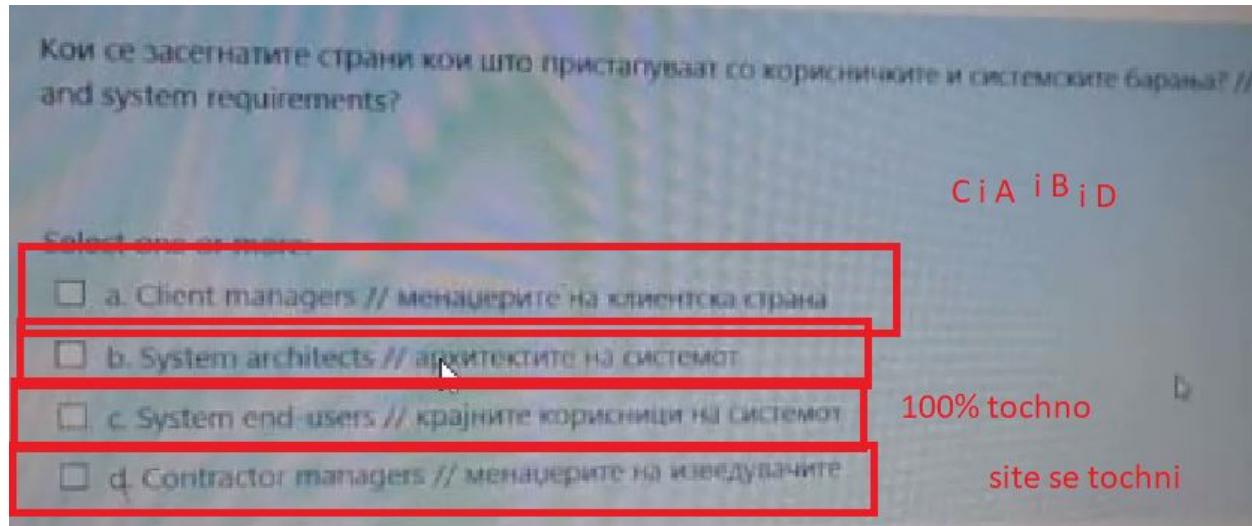
Agriculture: A \$2.4 Trillion Industry

FARMING PROVIDES JOBS FOR
1.3 BILLION

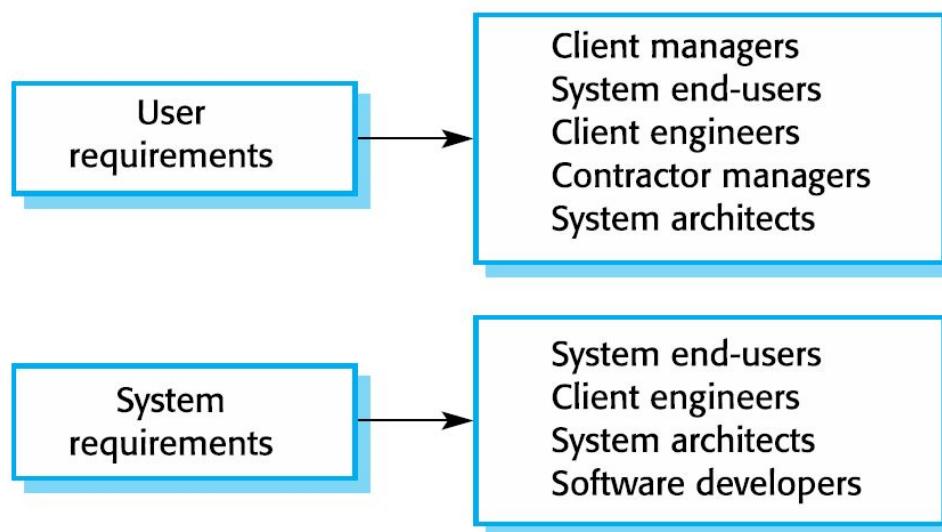


THE GLOBAL VALUE OF AGRICULTURE ADDS UP TO
\$2.4 TRILLION

6. Кои се засегнатите страни кои што пристапуваат со корисничките и системските барања?
- Менаџери на клиентска страна, Архитектите на системот, Крајните корисници на системот и менаџерите на изведувачите. - SE BARA PRESEK B I C SE TOCHNI!



VAKA KJE BESHE AKO E SO ILI... TOCHNO E POD B I C SAMO!!!!



user and system requirements
Ако се бара пресек: B I C.

7. Дадена е класата „SeptemberFirstExam“ со методот „public static boolean checkPrime(int n)“, каде врз основа на внесениот број, се враќа дали бројот е прост. За истиот метод е даден и Control Flow Graph.

Дадена е класата „SeptemberFirstExam“ со методот „public static boolean checkPrime(int n)“, каде врз основа на внесениот број, се враќа дали бројот е прост. За истиот метод е даден и Control Flow Graph.

There is a class "SeptemberFirstExam" with a method "public static boolean checkPrime(int n)", where based on the entered number, returns if the number is prime. For the same method, there is a Control Flow Graph.

```
public class SeptemberFirstExam {  
    public static boolean checkPrime(int n) {  
        int m = n / 2; //1  
        if (n == 0 || n == 1) { //2  
            return false; //3  
        } else {  
            for (int i = 2; i <= m; i++) { //4  
                if (n % i == 0) { //5  
                    return false; //6  
                }  
            }  
        }  
        return true; //7  
    } //8
```

Одговорете на следните прашања: / Answer the following questions:

Која е цикломатската комплексност? / What is the cyclomatic complexity?

Кој е минималниот број на тестови за „multiple condition“ методата за условот „if (n == 0 || n == 1)“? /

4 3

17:00 ENG 77% C K1 D1

Цикломатска комплексност: 4

Минимален број на тестови за „multiple condition“ методата за условот „if (n == 0 || n == 1)“? Izgleda e 3 bidejki za logichko ili vo 3 sluchai e tochno...

Која е цикломатската комплексност? / What is the cyclomatic complexity? 4

Кој е минималниот број на тестови за „multiple condition“ методата за условот „if ($n == 0 \parallel n == 1$)“?
/

What is the minimum number of tests for the "multiple condition" method for the condition "if ($n == 0 \parallel n == 1$)"?
◆

Доколку сакаме да напишеме метод за тестирање на **патеката 1-2-3-8** со користење на JUnit5, како би изгледала тест класата?
/ If we want to write a method for testing the **path 1-2-3-8** using JUnit5, what would the test class look like?

Не треба овде анотација / There should be no annotation here ◆

```
public class TestClass {  
  
    @Test  
    void checkPrimeTest() {  
        //Одберете ги валидните тест методи //Check the valid test methods  
         "assertFalse(SeptemberFirstExam.checkPrime(0));"  
         "assertTrue(SeptemberFirstExam.checkPrime(0));"  
         "assertFalse(SeptemberFirstExam.checkPrime(3));"  
         "assertTrue(SeptemberFirstExam.checkPrime(3));"  
         "assertEquals(SeptemberFirstExam.checkPrime(0), true);"  
         "assertEquals(SeptemberFirstExam.checkPrime(0), false);"  
         "assertEquals(SeptemberFirstExam.checkPrime(3), true);"  
         "assertEquals(SeptemberFirstExam.checkPrime(3), false);"  
    }  
}
```

2 3 5 8

17:14 ENG ⌂ ⏪ ⏴ ⏵ ⏵

so crveno e točno

a točno e zato što se testira pateka 1 2 3 8, spored kodot testovite za taa pateka ke bidat

točni ako se testira so 0 ili 1 (pogledni grafot 1 2 3 8)

testovite deka ima ponudeno 3 se false

zato što 3 ne pominuva niz taa pateka, treba da bide 0 ili 1

*Doobjasnuvanje:

ako se bara konkretно primer primerce za

konektiranje so baza na podatoci

beforeall kje stavish za konekcija

afterall za diskonektiranje

8. Во рамките на работите кои ги специфицираат, описите на процеси типично НЕ специфицираат: - Вештини - потребни за изведување на активност.

Во рамките на работите кои ги специфицираат, описите на процеси типично НЕ специфицираат: / Among many things, process descriptions usually do NOT specify:

Select one:

- a. Вештини - потребни за изведување на активност / Skills – required to perform an activity
- b. Редослед на активностите / Ordering of activities
- c. Роли (улоги) кои ги отсликуваат одговорностите на лицата инволвирали во процесот / Roles, which reflect the responsibilities of the people involved in the process
- d. Продукти кои се резултат на процесните активности / Products, which are the outcomes of a process activity

9. Внесете ја точната секвенца од извршување на акциите во рамки на дијаграмот на активности прикажан на сликата при извршување на следното корисничко сценарио: „Корисникот внесува корисничко име кое не постои во системот и лозинка“. Секвенцата треба да биде напишана во формат „123456789“, односно сите броеви да се споени (без празни места помеѓу нив). - 12349

Внесете ја точната секвенца од извршување на акциите во рамки на дијаграмот на активности прикажан на сликата при извршување на следното корисничко сценарио: „Корисникот внесува корисничко име кое не постои во системот и лозинка“. Секвенцата треба да биде напишана во формат "123456789", односно сите броеви да се споени (без празни места помеѓу нив). // Write the correct sequence of action execution from the activity diagram shown on the figure, when the user scenario "The user enters an user name that does not exist in the system and a password" is executed. The sequence should be written in the format "123456789" i.e there should be no spaces between the numbers.

```

graph TD
    Start((1)) --> EnterName([Внес на корисничко име])
    EnterName --> Decision1{Дали корисничкото име постои во системот?}
    Decision1 -- Не --> Error[Приказ на грешка]
    Decision1 -- Да --> EnterPassword([Внес на лозинка])
    EnterPassword --> Decision2{Дали лозинката се совпаѓа со корисничкото име?}
    Decision2 -- Не --> Error
    Decision2 -- Да --> Notification([Најава])
    Notification --> Decision3{9}
  
```

Answer: **12349**

10. ____ го спојува јазот помеѓу описот на системско ниво и дизајнот на софтверот.

- Analysis model

____ го спојува јазот помеѓу описот на системско ниво и дизајнот на софтверот. / ____ bridges the gap between a system level description and a software design.

Select one:

- a. Analysis model
- b. Requirement engineering
- c. Implementation
- d. Testing

A

11. SLIKI OD UVID:

Одговорете на следните прашања. / Answer the following questions.

Кој е минималниот број на тестови ком треба да се напишат за да се исполни условот за:
/ What is the minimum number of test cases that need to be written to meet the requirement for:

- C0 - Every statement методата / C0 - Every statement method
- C1 - Every branch методата / C1 - Every branch method

Кој е минималниот број на тестови за „multiple condition“ методата за условот „if ($n == 0 \parallel n == 1$)“?
/ What is the minimum number of tests for the "multiple condition" method for the condition "if ($n == 0 \parallel n == 1$)"?

+ X

Доколку скаме да напишеме метод за тестирање на патеката 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 и дека заврши овој тест?
/ If we want to write a method for testing the path 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 and that it is finishing for less

Tekst

Нека е даден проект SI со само една класа напишана во Java во него. По додавање на кодот (даден подолу) во оваа класа се прави commit. // Imagine that there is a project named SI with only one Java class in it. After the code (given below) is added to the class, the change is committed.

```
public class SISeptember {  
    int a = 6;  
    int b = 7;  
  
    public SISeptember(int a, int b) {  
        this.a = a;  
        this.b = b;  
    }  
}
```

Изберете ја командата со која што директориумот SI ќе го направите да биде локалниот git репозиториум: / Select the command that you'll use in order to make the directory SI a local Git repository:

git init ▼ ✓

Изберете ја командата со која што локалниот репозиториум ќе го поврзе со remote репозиториумот што се наоѓа на линкот https://github.com/test/test_repo.git // Select the command that you'll use to connect the local repo to a remote repo that's located on the link https://github.com/test/test_repo.git: git remote add origin https://github.com/test/test_repo.git ▼ ✓

Изберете ја командата со која ќе креирате и веднаш ќе се префрлите на гранката numbers // Select the command that you'll use to create and checkout immediately to the branch named numbers:

git checkout -b numbers ▼ ✓

Откако ќе се префрлите на numbers ја менувате вредноста на променливата b во 10, правите git add и git commit. Изберете ја командата со која ќе се префрлите на главната гранка // After checking out to the branch numbers you've changed the value of the variable b to 10 and you've executed git add and git commit. Select the command that you'll use to checkout to the main branch:

git checkout master ▼ ✓

Додека сте во главната гранка правите промена во класата и истата го добива следниот изглед. За промената соодветно правите commit: // While you are on the main branch you are making the following change. You've committed the corresponding change:

```
public class SISeptember {  
    int a = 6;  
    int b = 7;  
    int c;  
  
    public SISeptember(int a, int b) {  
        this.a = a;  
        this.b = b;  
        this.c = a+b;  
    }  
}
```

Изберете ја командата со која што ќе ја споите гранката numbers кон главната гранка. // Select the command that you'll use in order to merge the numbers branch into the main branch:

git merge numbers ▼ ✓

Спојувањето на гранката numbers кон гранката master: // The merging of the numbers towards the master branch: ќе резултира со конфликт//conflict will emerge ▼ ✓

Спојувањето на гранката numbers кон главната гранка ќе се изврши по стратегија: // The merging of the branch numbers towards the branch master will be done with the strategy:

нема да се изврши, ќе се јави конфликт//the merging won't be successful, conflict will emerge ▼ ✓

isplit.finki.ukim.mk/mod/quiz/reviewquestion.php?attempt=62138&slot=0

Monitoring Kafka... Confluent Platform... Docker With Spring... Spring Boot, MySQL...

Теодор Ангеловски

Quiz Втор дел - квик 2
Question new feature
Created on Monday, 14 September 2020, 10:55 AM

Работите во софтверска компанија која што го употребува **Gitflow workflow** моделот за колаборација на проект. Добивате нова задача да развиете нова страница за регистрација на ВИП корисниците на вашата апликација. Подредете ги git командите како што би ги употребиле за да се справите со овој предизвик. Се претпоставува дека решението ќе биде брзо и едноставно и ќе ви биде доволен еден commit. Моментално се налѓате на гранката **master**. Користете една команда за креирање и префрлувanje на нова гранка за решението да биде во 6 чекори! // You're working for a software company that uses Gitflow workflow. You've assigned a task to develop a new web page for registration of VIP users on your application. Sort the git commands that you would use in order to complete your task. It is assumed that the solution of the task will be simple and fast and you'll need only one git commit for it. At the moment you are on the master branch. Use only one command to create and checkout to a new branch, so that the solution can be in 6 steps!

1	git checkout develop	✓
2	git checkout -b release	✗
3	git add & git commit -m "some message"	✓
4	git checkout develop	✓
5	git merge --no-ff feature	✓
6	git branch -d feature	✓

Give your reasons

Правит чекор е да се префрлите на гранката **develop**. На оваа гранка ги има најновите промени кои не се дел од некоја **release** верзија на софтверскиот проект.

Следниот чекор е да креираме нова гранка и потоа да се префрлите на истата. Со наредбата **git checkout -b feature** се креира нова **feature** гранка и потоа се префрламе на неа. Наредбата **git checkout** е за префрлта на друга гранка, но аргументот **-b** е искористен за да се креира новата гранка бидејќи истата не постои.

Сега можеме да го завршиме нашиот таск и да ја креираме веб страницата за регистрација на ВИП корисници.

Откако ќе завршиме со креирањето треба новата датотека/датотеки да ги додадеме во **staging area** и потоа да направиме **commit**.

Се враќаме на гранката **develop**.

Правиме **merge** користејќи стратегија без **fast forward**.

Откако оваа гранка е споена, можеме да ја избрисаме **feature** гранката бидејќи таа веќе е непотребна.

Your answer is partially correct.

You have correctly selected 5.

The correct answer is: 1 → git checkout develop, 2 → git checkout -b feature, 3 → git add & git commit -m "some message", 4 → git checkout develop, 5 → git merge --no-ff feature, 6 → git branch -d feature

Работите во софтверска компанија која што го употребува **Gitflow workflow** моделот за колаборација на проект. Добивате нова апликација. Подредете ги git командите како што би ги употребиле за да се справите со овој предизвик. Се претпоставува дека commit. Моментално се наоѓате на гранката master. Користете една команда за креирање и префрлување на нова гранка за реш workflow. You're assigned a task to develop a new web page for registration of VIP users on your application. Sort the git commands that will be simple and fast and you'll need only one git commit for it. At the moment you are on the master branch. Use only one command to

- | | | |
|---|--|---|
| 1 | git checkout develop | ✓ |
| 2 | git checkout -b feature | ✓ |
| 3 | git add & git commit -m "some message" | ✓ |
| 4 | git checkout develop | ✓ |
| 5 | git merge --no-ff release | ✗ |
| 6 | git branch -d release | ✗ |

Give your reasons

Според Gitflow workflow кога развиваме нова задача не треба да работиме директно на мастер, туку веќе има предодредени правила доколку сакаме да додадеме нов feature ја разгрунуваме гранката девелоп и таму работиме, откако завршуваме со feature вклучиме феатурот во новата верзија.

Your answer is partially correct.

You have correctly selected 4.

The correct answer is: 1 → git checkout develop, 2 → git checkout -b feature, 3 → git add & git commit - m "some message", 4 → git c

We recommend drawing the diagram on a paper and then answer the questions.

Класниот дијаграм ќе има само класа Паркинг која ќе се однесува на двата типови паркинзи.

/

The class diagram will compose only of class Parking_Lot that represents both types of parking lots.

Неточно / False 

Correct

The correct answer is: Неточно / False - и класите Катна_Гаража и Зонско_Паркирање?

Mark 1.00 out of 1.00

With what type of relationship the classes MS_Garage and Zonal_Parking will be connected?

Со која класа ќе биде поврзана класата Паркинг_Место?

/

With which class the class Parking_Place will be connected?

Со каква врска ќе бидат поврзани класите Вработен и Паркинг?

/

With what type of relationship the classes Employee and Parking_Lot will be connected?

Со каква врска ќе бидат поврзани класите Вработен и Паркинг_Место?

/

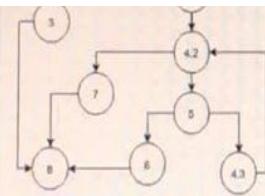
With what type of relationship the classes Employee and Parking_Place will be connected?

Која ќе биде кардиналноста на врската помеѓу класите Вработен и Паркинг_Место, од страната на класата Вработен?

/

```
if (n >= 0) {  
    if (n == 0) {  
        //...  
    } else if (n == 1) {  
        //...  
    } else {  
        //...  
    }  
}
```



Одговорете на следните прашања: / Answer the following questions:

Кој е минималниот број на тестови кои треба да се напишат за да се исполни условот за:
/ What is the minimum number of test cases that need to be written to meet the requirement for:

- C0 - Every statement методата / C0 - Every statement method

- C1 - Every branch методата / C1 - Every branch method

Кој е минималниот број на тестови за „multiple condition“ методата за условот „If ($n == 0 \text{ || } n == 1$)“?

What is the minimum number of tests for the "multiple condition" method for the condition "If ($n == 0 \text{ || } n == 1$)"?

Доколку сакамо да напишеме метод за тестирање на патеката 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 и дека завршува за помалку од 200 милисекунди, со користење на JUnit5, кој од следните методи е валиден за овој тест?

If we want to write a method for testing the path 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 and that it is finishing for less than 200 milliseconds using JUnit5, which of the following methods is valid for this test?

```
1. @Test  
    public void testMethod() {  
        long start = System.currentTimeMillis();  
        //...  
        long end = System.currentTimeMillis();  
        assertEquals(200, end - start);  
    }
```



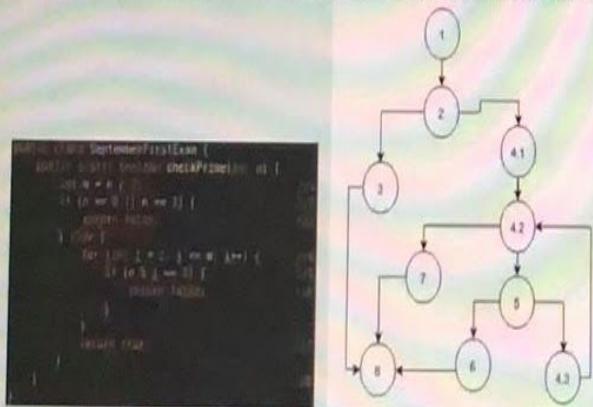
```
2. @Test  
    public void testMethod() {  
        long start = System.currentTimeMillis();  
        //...  
        long end = System.currentTimeMillis();  
        assertEquals(200, end - start);  
    }
```



```
3. @Test  
    public void testMethod() {  
        long start = System.currentTimeMillis();  
        //...  
        long end = System.currentTimeMillis();  
        assertEquals(200, end - start);  
    }
```

Дадена е класата „SeptemberFirstExam“ со методот „public static boolean checkPrime(int n)“, каде врз основа на внесениот број, се враќа дали бројот е прост. За истиот метод, е даден и Control Flow Graph.

/
There is a class "SeptemberFirstExam" with a method "public static boolean checkPrime(int n)", where based on the entered number, returns if the number is prime. For the same method, there is a Control Flow Graph.



Одговорете на следните прашања: / Answer the following questions:

Кој е минималниот број на тестови кои треба да се напишат за да се исполнi условот за:

/ What is the minimum number of test cases that need to be written to meet the requirement for:

- C0 - Every statement методата / C0 - Every statement method



- C1 - Every branch методата / C1 - Every branch method



Кој е минималниот број на тестови за „multiple condition“ методата за условот „if (n == 0 || n == 1)“?

/

What is the minimum number of tests for the "multiple condition" method for the condition "if (n == 0 || n == 1)"?



Дошолку сакаме да напишеме метод за тестирање на патеката 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 и дека завршува за помалку од 200 милисекунди, со користење на JUnit5, кој од следните методи е валиден за овој тест?

If we want to write a method for testing this path 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 and that it's finished for less than 200 milliseconds using JUnit5, which of the following methods is valid for this test?

1 Monday, 14 September 2015, 10:00

Работите во софтверска компанија која што го употребува Gitflow workflow моделот за колаборација на проект. Добивате нова звезда да развиете нова страница за регистрација на VIP кориснициште на вашата апликација. Подредете ги git командите како што би ги употребиле за да се справите со овој предизвик. Се претпоставува дека решението ќе биде брзо и едноставно и ќе ви биде доволен еден commit. Моментално се наоѓате на гранката master. Користете една команда за креирање и префлувување на нова гранка за решението да биде во 6 чекори! // You're working for a software company that uses Gitflow workflow. You're assigned a task to develop a new web page for registration of VIP users on your application. Sort the git commands that you would use in order to complete your task. It is assumed that the solution of the task will be simple and fast and you'll need only one git commit for it. At the moment you are on the master branch. Use only one command to create and checkout to a new branch, so that the solution can be in 6 steps!

- | | | |
|---|--|---|
| 1 | git checkout develop | ✓ |
| 2 | git checkout -b feature | ✓ |
| 3 | git add & git commit -m "some message" | ✓ |
| 4 | git checkout develop | ✓ |
| 5 | git merge --no-ff release | ✗ |
| 6 | git branch -d release | ✗ |

Give your reasons

Според Gitflow workflow кога развишаме нова задача не треба да работиме директно на мастер, тука веќе има предодредено гранки каде што тоа треба да се случува. Развиваме на гранката девелоп, а доколку сакаме да додадеме нов feature ја разгрануваме гранката девелоп и таму работиме, откако завршувааме со featureот го спојуваме со гранката девелоп, тестираме и одлучуваме дали ќе го вклучиме феатурот во новата верзија.

Your answer is partially correct.

You have correctly selected 4.

The correct answer is: 1 → git checkout develop, 2 → git checkout -b feature, 3 → git add & git commit - m "some message", 4 → git checkout develop, 5 → git merge --no-ff feature, 6 → git branch -d feature

Make comment or override mark

What is the minimum number of tests for the "multiple condition" method for the condition "if (n == 0 || n == 1)"?

Доколку сакаме да напишеме метод за тестирање на патеката 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 и дека завршува за помалку од 200 милисекунди, со користење на JUnit5, кој од следните методи за овој тест?

/ If we want to write a method for testing the path 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 and that it is finishing for less than 200 milliseconds using JUnit5, which of the following methods is valid for this test?

```
class TestMethod1 {
    @Test
    void testMethod1() {
        assertEquals("Method 1", calculatePath(1));
        assertEquals("Method 1", calculatePath(2));
        assertEquals("Method 1", calculatePath(4.1));
        assertEquals("Method 1", calculatePath(4.2));
        assertEquals("Method 1", calculatePath(5));
        assertEquals("Method 1", calculatePath(4.3));
        assertEquals("Method 1", calculatePath(6));
        assertEquals("Method 1", calculatePath(8));
    }
}

class TestMethod2 {
    @Test
    void testMethod2() {
        assertEquals("Method 2", calculatePath(1));
        assertEquals("Method 2", calculatePath(2));
        assertEquals("Method 2", calculatePath(4.1));
        assertEquals("Method 2", calculatePath(4.2));
        assertEquals("Method 2", calculatePath(5));
        assertEquals("Method 2", calculatePath(4.3));
        assertEquals("Method 2", calculatePath(6));
        assertEquals("Method 2", calculatePath(8));
    }
}

class TestMethod3 {
    @Test
    void testMethod3() {
        assertEquals("Method 3", calculatePath(1));
        assertEquals("Method 3", calculatePath(2));
        assertEquals("Method 3", calculatePath(4.1));
        assertEquals("Method 3", calculatePath(4.2));
        assertEquals("Method 3", calculatePath(5));
        assertEquals("Method 3", calculatePath(4.3));
        assertEquals("Method 3", calculatePath(6));
        assertEquals("Method 3", calculatePath(8));
    }
}

class TestMethod4 {
    @Test
    void testMethod4() {
        assertEquals("Method 4", calculatePath(1));
        assertEquals("Method 4", calculatePath(2));
        assertEquals("Method 4", calculatePath(4.1));
        assertEquals("Method 4", calculatePath(4.2));
        assertEquals("Method 4", calculatePath(5));
        assertEquals("Method 4", calculatePath(4.3));
        assertEquals("Method 4", calculatePath(6));
        assertEquals("Method 4", calculatePath(8));
    }
}
```

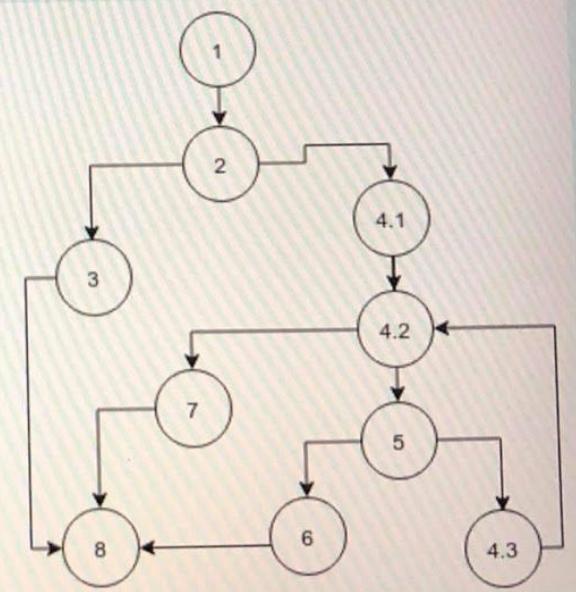
Give your reasons

c0 | c1 ги одредуваме според бројот на патеки, во ц1 постојат 3 опции да не влезе во циклусот, да влезе и да заврши и да влезе и да прекине со тоа ги поминувамо сите стечјменти. Исто тоа важи и за ц1 |

границите со 3 теста да се посетат еднаш тестот ке е 0 , втор тест ке е прост број , и трет ке е било кој број што не е прост.

максималниот блок на тестови за "multiple condition" методот за условот "if (n == 0 || n == 1)" е 2 билејќи има или редакција а тоа значи нама да е исполнето само кога и двете не се исполнуваат.

```
public class SeptemberFirstExam {
    public static boolean checkPrime(int n) {
        int m = n / 2; //1
        if (n == 0 || n == 1) { //2
            return false; //3
        } else {
            for (int i = 2; i <= m; i++) { //4
                if (n % i == 0) { //5
                    return false; //6
                }
            }
            return true; //7
        }
    }
}
```



Одговорете на следните прашања: / Answer the following questions:

ing Kafka...

Confluent Platfor...

Docker With Spring...

Spring Boot, MySQL...

/ If we want to write a method for testing the path 1 - 2 - 4.1 - (4.2 - 5 - 4.3) - 4.2 - 5 - 6 - 8 and that it is finishing for less than 200 milliseconds using JUnit5, which of the following methods is valid for this test?

```
public void testMethod0() {
    assertEquals(SegmentFirstOneFirstOne.checkPoint == 0);
    assertEquals(IterationTime100ms(200), 0 > SegmentFirstOneFirstOne.checkPoint >= 0);
}

public void testMethod1() {
    assertEquals(SegmentFirstOneFirstOne.checkPoint == 0);
    assertEquals(IterationTime100ms(200), 0 > SegmentFirstOneFirstOne.checkPoint >= 0);
}

public void testMethod2() {
    assertEquals(SegmentFirstOneFirstOne.checkPoint == 0);
    assertEquals(IterationTime100ms(200), 0 > SegmentFirstOneFirstOne.checkPoint >= 0);
}

public void testMethod3() {
    assertEquals(SegmentFirstOneFirstOne.checkPoint == 0);
    assertEquals(IterationTime100ms(200), 0 > SegmentFirstOneFirstOne.checkPoint >= 0);
}

public void testMethod4() {
    assertEquals(SegmentFirstOneFirstOne.checkPoint == 0);
    assertEquals(IterationTime100ms(200), 0 > SegmentFirstOneFirstOne.checkPoint >= 0);
}

public void testMethod5() {
    assertEquals(SegmentFirstOneFirstOne.checkPoint == 0);
    assertEquals(IterationTime100ms(200), 0 > SegmentFirstOneFirstOne.checkPoint >= 0);
}

public void testMethod6() {
    assertEquals(SegmentFirstOneFirstOne.checkPoint == 0);
    assertEquals(IterationTime100ms(200), 0 > SegmentFirstOneFirstOne.checkPoint >= 0);
}
```

testMethod4



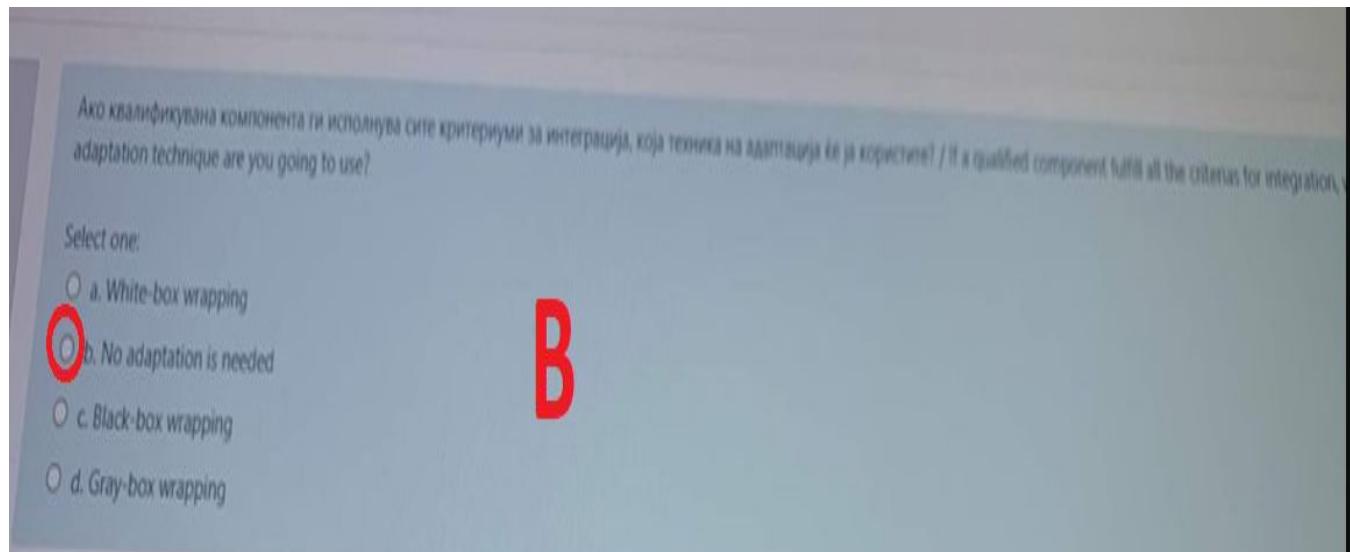
Give your reasons

Со и с тие одредуваме според бројот на патеки, во тој постојат 3 опции да не влезе во циклусот, да влезе и да заврши и да влезе и да прекине со тоа ги поминуваме сите стейменти. Исто тоа важи и за тој може сите граници со 3 теста да се посетат еднаш тестот кога е 0, втор тест кога е прост број, и трет кога е било кој број што не е прост. минималниот број на тестови за „multiple condition“ методата за условот „if (n == 0 || n == 1)“ е 2 бидејќи има или релација таа значи нема да е исполнето само кога и двете нема да го исполнуваат условот во секој друг случај ќе биде точна.

Make comment or override mark

View history

12. Ако квалификувана компонента ги исполнува критериуми за интеграција, која техника на адаптација ќе ја користите? - No adaption needed.



The rationale of CBD is that a component is an independent reuse unit for the construction of an application. However, at present there are not many components that can be actually reused without being adapted. The problem is caused by for example, incomplete component specifications, the mismatches between components and the reuse context including the application architecture, required functionality, software and hardware environments, quality attributes and other collaborating components. To tackle this problem, it is desirable to provide an automated deep component adaptation technology. Deep" adaptation is understood to mean an adaptation of the structure of a component, that is, an adaptation that goes beyond a simple conversion of the inputs to or outputs from the component that treats the component as a "black box" with no knowledge of the functionality of the component.

13. Информациите на корисничкиот интерфејс треба да бидат презентирани прогресивно (постепено)

Информациите на корисничкиот интерфејс треба да бидат презентирани _____ / The informations on the user interface should be presented in a _____

Select one:

a. прогресивно (постепено) / progressive fashion

b. комплетно / complete form

c. оформена целост / meaningful form

d. агресивно / aggressive fashion

[Clear my choice](#)

Progressive disclosure

A

14. Класен УМЛ (UML Class Diagram) / Klasen

UML Class Diagram

- * Components:
 1. Name, 2. Attributes, 3. Methods, 4. Relationships.
- * Visibility: (+)-public, (-)-private, (#)-protected, (v)-default
- * Relationships:
 1. Dependency $\dashv \rightarrow$ Најчеста врста вршејќи соодветјење. Ќе симболизира дека една класа има потреба од друга класа и може да го користи неговите објекти.
 2. Association — осовина врска од dependency. Ќе се наслика дека една класа има потреба од некоја сојузница од врски.
 3. Aggregation \diamond јаснина врска од association и се користи да покаже дека една класа има потреба да користи објекти од другата класа, но ако тие не се губат од другата класа.
 4. Composition \blacklozenge јаснина врска од aggregation каде се покажува имплементација на посебни начини.
- * Inheritance \rightarrow најчеста врска на врска и се користи да покаже класа која предава деловите на друга класа.
- * Најчеста од најчести: generalization, composition, aggregation, association, dependency (strongest to weakest).
- * Задача: да покажете како која е **抽象на** единица каде да се даде некоја конкретна имплементација. Тоа може да је најчеста врска када ќе дефинирате некоја нова класа када некоја не биде во **��е**.

Size	Abstract class name and operations in italic
Notes	
References	

15.

Tekst

Edited on Saturday, 12 September 2020, 5:15 PM

Како ќе го моделирате следното сценарио користејќи UML Use Case дијаграм?

Механичар прави сервис за автомобил. За време на сервисот, можеби ќе биде потребно да се сменуј единицата за кочење.

How do you model the following situation using UML Use Case diagram?
A mechanic does a car service. During that service, it might be necessary to change the break unit.

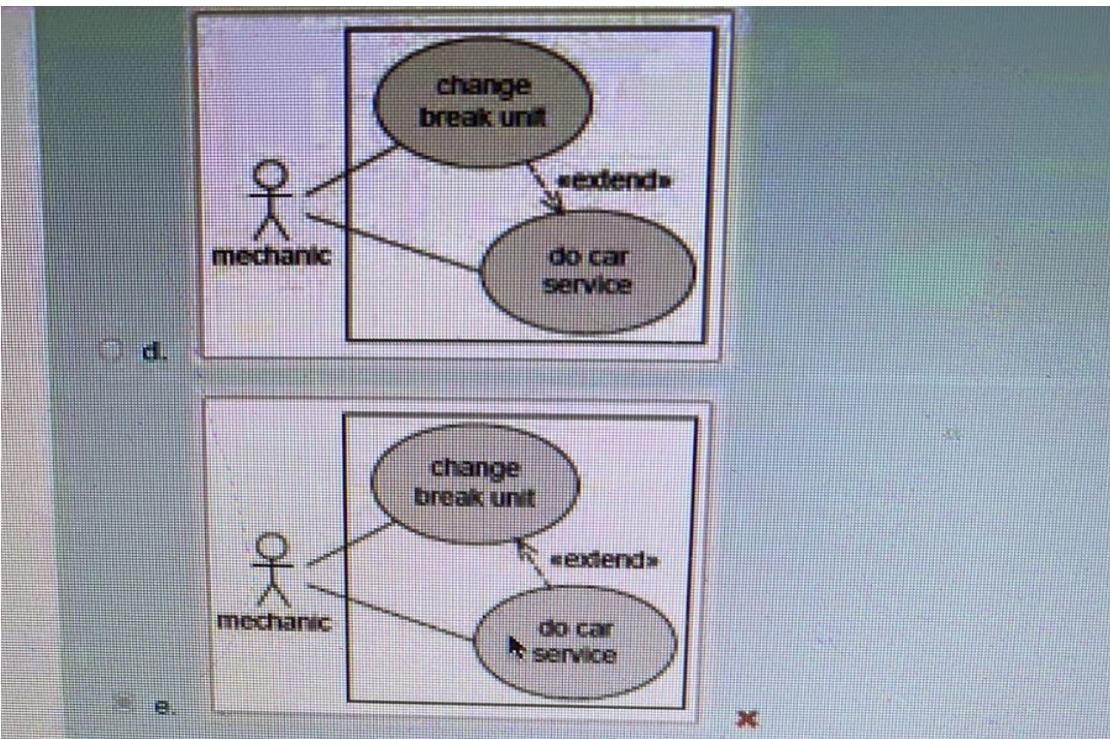
Select one:

a.

```
graph LR; Mechanic((Mechanic)) --> DoCarService{do car service}; subgraph " " [ ]; DoCarService --> ChangeBreakUnit{change break unit}; end
```

b.

```
graph LR; Mechanic((Mechanic)) --> DoCarService{do car service}; subgraph " " [ ]; DoCarService --> ChangeBreakUnit{change break unit}; end; Note["<<include>>"] --- AssociationLine[ ]
```



Your answer is incorrect.

The correct answer is:

