1. In Programarea Orientata-Obiect, un obiect este:
    a. o forma geometrica
    b. o lista de instructiuni
    c. o instanta a unei clase
    d. un tip specific de array
2. Care dintre urmatorii sunt specificatori de acces valizi in Java?
    a. public
    b. private
    c. void
    d. final
3. Metodele interfetei SpaceShip sunt implicit:
    a. public
    b. private
    c. default
    d. static
4. In codul problemei sunt definite:
    a. 2 interfete, 7 clase, 2 clase interne anonime
    b. 2 interfete, 7 clase
    c. 2 interfete, 6 clase, 2 clase interne anonime
    d. 2 interfete, 6 clase
5. Supraincarcarea este:
    a. Redefinirea unei metode intr-o subclasa cu aceiasi parametrii ca in superclasa
    b. Definirea unei metode de mai multe ori intr-o clasa, cu liste diferite de parametrii
    c. Alocarea dinamica a unei zone de memorie
    d. Refolosirea unei zone de memorie dupa ce a fost eliberata de Garbage Collector
6. Adnotarea @Override a metodei `avoid` arata ca:
    a. Metoda este supraincarcata
    b. Metoda trebuie apelata in program
    c. Metoda suprascrie o metoda din superclasa
    d. Metoda trebuie suprascrisa in subclase
7. Care dintre urmatoarele afirmatii este corecta?
    a. Compunerea este o relatie mai puternica decat Agregarea
    b. Agregarea este o relatie de tip HAS-A
    c. Asocierea este o relatie mai puternica decat Agregarea
    d. Mostenirea este o relatie de tip HAS-A
8. Relatia dintre Shuttle si SpaceShip e de:
    a. Asociere
    b. Agregare
    c. Compunere
    d. Mostenire
9. `(Asteroid)spaceRock` este un exemplu de:
    a. Upcast
    b. Downcast
    c. Conversion
    d. Copy prin CopyConstructor
10. O subclasa aflata in acelasi pachet cu superclasa ei mosteneste:
    a. toate campurile si metodele public
    b. toate campurile si metodele private
    c. toate campurile si metodele protected
    d. toate campurile si metodele default
11. O clasa obisnuita poate extinde o clasa abstracta:
    a. Daca nu are nicio metoda statica
    b. Daca sunt in acelasi pachet
    c. Daca ii implementeaza toate metodele abstracte
    d. Daca suprascrie cel putin o metoda a clasei abstracte.
12. Daca am dori sa adaugam clasa LiquidFuelRocket, aceasta ar avea sens:
    a. Sa extinda clasa Shuttle
    b. Sa fie compusa in clasa Shuttle
    c. Sa extinda clasa Rocket
    d. Sa fie compusa in clasa Rocket
13. Problema diamant apare cand doua clase B si C mostenesc o clasa A, si o alta clasa D mosteneste atat B, cat si C, intrucat:
    a. Daca D suprascrie o metoda din B, pe care C nu o are, rezulta un comportament nedeterminat
    b. Daca o metoda in A e suprascrisa de B si C, fara ca D sa o suprascrie, este ambiguu pe care implementare o va mosteni
    c. Daca B si C trebuie sa suprascrie o metoda din A in acelasi fel se duplica cod
    d. Nu putem face ca C si D sa suprascrie o metoda din A, dar B nu.
14. Ce obiecte vor fi create pentru urmatorul input:
```
Shuttle
asteroid
```
    a. un obiect Shuttle si unul Asteroid
    b. un obiect Rocket si unul Asteroid
    c. un obiect Shuttle si unul Meteor
    d. un obiect Rocket si unul Meteor
15. La curs am discutat despre polimorfism:
    a. Static
    b. Dinamic
    c. Complex
    d. Parametric
16. Faptul ca in cod se apeleaza avoid pentru referinte de tip SpaceShip, care de fapt se refera la obiecte Shuttle si Rocket, este o folosire a polimorfismului:
    a. Static
    b. Dinamic
    c. Complex
    d. Parametric
17. O clasa interna anonima:

a. Poate sa extinda o singura clasa sau sa implementeze o singura interfata
b. Nu are constructori proprii
c. Este instantiata intr-un singur loc
d. Trebuie sa extinda java.lang.Anonymous

18. Care dintre urmatoarele este superclasa tuturor erorilor si exceptiilor din Java?
    a. RunTimeException
    b. Throwable
    c. Catchable
    d. Exception
19. Care principiu de design orientat obiect militeaza ca o clasa nu ar trebui fortata sa implementeze metode pe care nu le va folosi?
    a. SRP
    b. LSP
    c. ISP
    d. DIP
20. LSP este reprezentat in codul de la problema prin:
    a. Metoda avoid este supraincarcata
    b. Se foloseste sintaxa try-catch pentru a asigura apelarea corecta a metodei avoid
    c. In main se poate apela avoid pentru obiectul SpaceShip indiferent de implementare, Rocket sau Shuttle
    d. Metoda create este statica
21. Care principiu de design orientat obiect este incalcat de clasa Solution prin faptul ca se ocupa de citire si apelul metodei avoid?
    a. SRP
    b. LSP
    c. ISP
    d. DIP
22. DIP este util intrucat, atunci cand este respectat:
    a. implementarile pot fi inlocuite fara a afecta codul care le foloseste
    b. o clasa nu este fortata sa implementeze metode care nu vor fi folosite
    c. obiectele dintr-un program pot fi inlocuite cu implementari ale subtipurilor
    d. reduce numarul de clase si interfete din program
23. Ce design pattern este folosit de clasa Scanner?
    a. Visitor
    b. Iterator
    c. Observer
    d. Builder
24. Care dintre urmatoarele se incadreaza in categoria de Behavioral Design Patterns:
    a. Singleton
    b. Visitor
    c. Observer
    d. Factory
25. In care pattern nu se poate aplica DIP?
    a. Observer
    b. Factory
    c. Singleton
    d. Visitor
26. Ce design pattern este folosit pentru a crea obiecte din ierarhia clasei SpaceRock in contextul problemei:
    a. Singleton
    b. Factory
    c. Builder
    d. Strategy
27. Inversion of Control este un principiu de design prin care:
    a. Tipurile folosite in definirea claselor sunt parametrizate
    b. Codul scris de dezvoltator este apelat de un framework generic
    c. Se alege la runtime care metoda sa fie apelata in functie de tipurile a doua obiecte
    d. Este inversata abordarea traditionala in care codul scris de dezvoltator apeleaza biblioteci reutilizabile
28. Care dintre urmatoarele tehnici poate fi aplicata pentru a pacali o implementare de Singleton bazata pe constructor privat?
    a. Mostenire
    b. Introspectie
    c. Reflectie
    d. Adnotare

PROBLEMA:

La testul practic, studentilor li s-a cerut sa completeze un program, scopul fiind sa arate ca au inteles conceptul de Double Dispatch, cu mentiunea ca orice folosire a operatorului `instanceof` va fi drastic penalizata.

Hacker din fire, unul dintre studenti a produs solutia din codul atasat, liniile scrise de el fiind marcate de comentarii. Haideti sa analizam un pic aceasta solutie:

1. Explicati de ce studentul a folosit try-catch.
2. Ce ar trebui modificat pentru a implementa corect Double Dispatch? Hint: interfata SpaceRock si clasele care o implementeaza nu ar trebui sa ramana goale, iar codul scris de student ar trebui inlocuit.
3. Comentati cu privire la extensibilitatea programului. Ce fel de modificari ar putea fi dorite si cat de usor sau greu ar putea fi implementate? Comparati, ca extensibilitate, implementarea studentului hacker si cea de la punctul 2.

1. In Programarea Orientata-Obiect, un obiect este:
    a. o forma geometrica
    b. o lista de instructiuni
    c. o instanta a unei clase
    d. un tip specific de array
2. Care dintre urmatorii sunt specificatori de acces valizi in Java?
    a. void
    b. public
    c. private
    d. final
3. Metodele interfetei SpaceShip sunt implicit:
    a. default
    b. public
    c. private
    d. static
4. Supraincarcarea este:
    a. Redefinirea unei metode intr-o subclasa cu aceiasi parametrii ca in superclasa
    b. Definirea unei metode de mai multe ori intr-o clasa, cu liste diferite de parametrii
    c. Alocarea dinamica a unei zone de memorie
    d. Refolosirea unei zone de memorie dupa ce a fost eliberata de Garbage Collector
5. Care dintre urmatoarele afirmatii este corecta?
    a. Compunerea este o relatie mai puternica decat Agregarea
    b. Asocierea este o relatie mai puternica decat Agregarea
    c. Agregarea este o relatie de tip HAS-A
    d. Mostenirea este o relatie de tip HAS-A
6. Adnotarea @Override a metodei `avoid` arata ca:
    a. Metoda este supraincarcata
    b. Metoda trebuie apelata in program
    c. Metoda trebuie suprascrisa in subclase
    d. Metoda suprascrie o metoda din superclasa
7. Relatia dintre Shuttle si SpaceShip e de:
    a. Mostenire
    b. Asociere
    c. Agregare
    d. Compunere
8. `(Asteroid)spaceRock` este un exemplu de:
    a. Downcast
    b. Upcast
    c. Conversion
    d. Copy prin CopyConstructor
9. O subclasa aflata in acelasi pachet cu superclasa ei mosteneste:
    a. toate campurile si metodele default
    b. toate campurile si metodele public
    c. toate campurile si metodele private
    d. toate campurile si metodele protected
10. O clasa obisnuita poate extinde o clasa abstracta:
    a. Daca nu are nicio metoda statica
    b. Daca ii implementeaza toate metodele abstracte
    c. Daca sunt in acelasi pachet
    d. Daca suprascrie cel putin o metoda a clasei abstracte.
11. Daca am dori sa adaugam clasa LiquidFuelRocket, aceasta ar avea sens:
    a. Sa extinda clasa Shuttle
    b. Sa extinda clasa Rocket
    c. Sa fie compusa in clasa Shuttle
    d. Sa fie compusa in clasa Rocket
12. Problema diamant apare cand doua clase B si C mostenesc o clasa A, si o alta clasa D mosteneste atat B, cat si C, intrucat:
    a. Daca o metoda in A e suprascrisa de B si C, fara ca D sa o suprascrie, este ambiguu pe care implementare o va mosteni
    b. Daca D suprascrie o metoda din B, pe care C nu o are, rezulta un comportament nedeterminat
    c. Daca B si C trebuie sa suprascrie o metoda din A in acelasi fel se duplica cod
    d. Nu putem face ca C si D sa suprascrie o metoda din A, dar B nu.
13. Ce obiecte vor fi create pentru urmatorul input:
```
Shuttle
asteroid
```
    a. un obiect Shuttle si unul Asteroid
    b. un obiect Shuttle si unul Meteor
    c. un obiect Rocket si unul Meteor
    d. un obiect Rocket si unul Asteroid
14. La curs am discutat despre polimorfism:
    a. Complex
    b. Static
    c. Dinamic
    d. Parametric
15. O clasa interna anonima:
    a. Poate sa extinda o singura clasa sau sa implementeze o singura interfata
    b. Nu are constructori proprii
    c. Este instantiata intr-un singur loc
    d. Trebuie sa extinda java.lang.Anonymous
16. Faptul ca in cod se apeleaza avoid pentru referinte de tip SpaceShip, care de fapt se refera la obiecte Shuttle si Rocket, este o folosire a polimorfismului:
    a. Static
    b. Dinamic
    c. Complex
    d. Parametric
17. Care dintre urmatoarele este superclasa tuturor erorilor si exceptiilor din Java?

a. RunTimeException
b. Catchable
c. Exception
d. Throwable

18. Care principiu de design orientat obiect militeaza ca o clasa nu ar trebui fortata sa implementeze metode pe care nu le va folosi?
    a. SRP
    b. ISP
    c. LSP
    d. DIP
19. LSP este reprezentat in codul de la problema prin:
    a. Metoda avoid este supraincarcata
    b. Metoda create este statica
    c. Se foloseste sintaxa try-catch pentru a asigura apelarea corecta a metodei avoid
    d. In main se poate apela avoid pentru obiectul SpaceShip indiferent de implementare, Rocket sau Shuttle
20. Care principiu de design orientat obiect este incalcat de clasa Solution prin faptul ca se ocupa de citire si apelul metodei avoid?
    a. DIP
    b. SRP
    c. LSP
    d. ISP
21. DIP este util intrucat, atunci cand este respectat:
    a. o clasa nu este fortata sa implementeze metode care nu vor fi folosite
    b. implementarile pot fi inlocuite fara a afecta codul care le foloseste
    c. obiectele dintr-un program pot fi inlocuite cu implementari ale subtipurilor
    d. reduce numarul de clase si interfete din program
22. Ce design pattern este folosit de clasa Scanner?
    a. Visitor
    b. Builder
    c. Iterator
    d. Observer
23. Care dintre urmatoarele se incadreaza in categoria de Behavioral Design Patterns:
    a. Visitor
    b. Singleton
    c. Observer
    d. Factory
24. In care pattern dintre urmatoarele nu se poate aplica DIP?
    a. Observer
    b. Singleton
    c. Factory
    d. Visitor
25. Ce design pattern este folosit pentru a crea obiecte din ierarhia clasei SpaceRock in contextul problemei:
    a. Singleton
    b. Factory
    c. Builder
    d. Strategy
26. In codul problemei sunt definite:
    a. 2 interfete, 7 clase, 2 clase interne anonime
    b. 2 interfete, 6 clase, 2 clase interne anonime
    c. 2 interfete, 7 clase
    d. 2 interfete, 6 clase
27. Inversion of Control este un principiu de design prin care:
    a. Este inversata abordarea traditionala in care codul scris de dezvoltator apeleaza biblioteci reutilizabile
    b. Tipurile folosite in definirea claselor sunt parametrizate
    c. Codul scris de dezvoltator este apelat de un framework generic
    d. Se alege la runtime care metoda sa fie apelata in functie de tipurile a doua obiecte
28. Care dintre urmatoarele tehnici poate fi aplicata pentru a pacali o implementare de Singleton bazata pe constructor privat?
    a. Reflectie
    b. Mostenire
    c. Introspectie
    d. Adnotare

PROBLEMA:

La testul practic, studentilor li s-a cerut sa completeze un program, scopul fiind sa arate ca au inteles conceptul de Double Dispatch, cu mentiunea ca orice folosire a operatorului `instanceof` va fi drastic penalizata.

Hacker din fire, unul dintre studenti a produs solutia din codul atasat, liniile scrise de el fiind marcate de comentarii. Haideti sa analizam un pic aceasta solutie:

1. Explicati de ce studentul a folosit try-catch.
2. Ce ar trebui modificat pentru a implementa corect Double Dispatch? Hint: interfata SpaceRock si clasele care o implementeaza nu ar trebui sa ramana goale, iar codul scris de student ar trebui inlocuit.
3. Comentati cu privire la extensibilitatea programului. Ce fel de modificari ar putea fi dorite si cat de usor sau greu ar putea fi implementate? Comparati, ca extensibilitate, implementarea studentului hacker si cea de la punctul 2.

Programare Orientata pe Obiecte, 7 februarie 2019

1. In Programarea Orientata-Obiect, un obiect este:
    a. o forma geometrica
    b. o lista de instructiuni
    c. o instanta a unei clase
    d. un tip specific de array
2. Care dintre urmatorii sunt specificatori de acces valizi in Java?
    a. public
    b. static
    c. void
    d. protected
3. Metodele interfetei SpaceShip sunt implicit:
    a. private
    b. default
    c. public
    d. static
4. In codul problemei sunt definite:
    a. 7 clase, 2 interfete, 1 clasa interne anonima
    b. 7 clase, 2 interfete, 2 clase interne anonime
    c. 7 clase, 2 interfete, 3 clase interne anonime
    d. 7 clase, 2 interfete
5. Suprascrierea este:
    a. Redefinirea unei metode intr-o subclasa cu aceiasi parametrii ca in superclasa
    b. Definirea unei metode de mai multe ori intr-o clasa cu liste diferite de parametrii
    c. Alocarea dinamica a unei zone de memorie
    d. Refolosirea unei zone de memorie dupa ce a fost eliberata de Garbage Collector
6. Adnotarea @Override a metodei `avoid` arata ca:
    a. Metoda este supraincarcata
    b. Metoda trebuie apelata in program
    c. Metoda suprascrie o metoda din superclasa
    d. Metoda trebuie suprascrisa in subclase
7. Care dintre urmatoarele afirmatii este corecta?
    a. Compunerea este o relatie mai puternica decat Agregarea
    b. Agregarea este o relatie de tip IS-A
    c. Asocierea este o relatie mai puternica decat Agregarea
    d. Mostenirea este o relatie de tip IS-A
8. Relatia dintre Asteroid si SpaceRock e de:
    a. Asociere
    b. Agregare
    c. Compunere
    d. Mostenire
9. `(Shuttle)spaceShip` este un exemplu de:
    a. Conversion
    b. Upcast
    c. Downcast
    d. Copy prin CopyConstructor
10. O subclasa aflata in acelasi pachet cu superclasa ei mosteneste:
    a. toate campurile si metodele public
    b. toate campurile si metodele protected
    c. toate campurile si metodele private
    d. toate campurile si metodele default
11. Cand se apeleaza `new Shuttle()` se va invoca:
    a. Un constructor fara parametrii din clasa Shuttle, care nu exista, deci avem eroare
    b. Un constructor fara parametrii pe care clasa Shuttle il mosteneste de la interfata SpaceShip
    c. Un constructor fara parametrii declarat implicit in clasa Shuttle
    d. Un constructor fara parametrii pe care clasa Shuttle il mosteneste de la clasa Solution
12. O clasa obisnuita poate extinde o clasa abstracta:
    a. Daca nu are nicio metoda statica
    b. Daca sunt in acelasi pachet
    c. Daca ii implementeaza toate metodele abstracte
    d. Daca suprascrie cel putin o metoda a clasei abstracte.
13. Daca am dori sa adaugam clasa SpaceXShuttle, aceasta ar avea sens:
    a. Sa extinda clasa Shuttle
    b. Sa fie compusa in clasa Shuttle
    c. Sa extinda clasa Rocket
    d. Sa fie compusa in clasa Rocket
14. Problema diamant apare cand doua clase B si C mostenesc o clasa A, si o alta clasa D mosteneste atat B, cat si C, intrucat:
    a. Daca D suprascrie o metoda din B, pe care C nu o are, rezulta un comportament nedeterminat
    b. Daca o metoda in A e suprascrisa de B si C, fara ca D sa o suprascrie, este ambiguu pe care implementare o va mosteni
    c. Daca B si C trebuie sa suprascrie o metoda din A in acelasi fel se duplica cod
    d. Nu putem face ca C si D sa suprascrie o metoda din A, dar B nu.
15. Ce obiecte vor fi create pentru urmatorul input:

```
shuttle
Asteroid
```

    a. un obiect Shuttle si unul Asteroid
    b. un obiect Rocket si unul Asteroid
    c. un obiect Shuttle si unul Meteor
    d. un obiect Rocket si unul Meteor
16. La curs am discutat despre polimorfism:
    a. Static
    b. Dinamic

c. Complex
d. Parametric

17. Faptul ca in cod se apeleaza avoid atat cu parametrii de tip Asteroid, cat si Meteor este o folosire a polimorfismului:
    a. Static
    b. Dinamic
    c. Complex
    d. Parametric
18. O clasa interna anonima:
    a. Poate sa extinda o singura clasa sau sa implementeze o singura interfata
    b. Nu are constructori proprii
    c. Este instantiata intr-un singur loc
    d. Trebuie sa extinda java.lang.Anonymous
19. Care dintre urmatoarele este superclasa tuturor erorilor si exceptiilor din Java?
    a. RunTimeException
    b. Throwable
    c. Catchable
    d. Exception
20. Care principiu de design orientat obiect militeaza ca o clasa ar trebui sa aiba un singur motiv pentru care poate fi schimbata?
    a. SRP
    b. LSP
    c. ISP
    d. DIP
21. LSP este reprezentat in codul de la problema prin:
    a. Metoda avoid este supraincarcata
    b. Se foloseste sintaxa try-catch pentru a asigura apelarea corecta a metodei avoid
    c. In main se poate apela avoid pentru obiectul SpaceShip indiferent de implementare, Rocket sau Shuttle
    d. Metoda create este statica
22. ISP este util intrucat, atunci cand este respectat:
    a. implementarile pot fi inlocuite fara a afecta codul care le foloseste
    b. o clasa nu este fortata sa implementeze metode care nu vor fi folosite
    c. obiectele dintr-un program pot fi inlocuite cu implementari ale subtipurilor
    d. reduce numarul de clase si interfete din program
23. Ce design pattern este folosit de clasa Scanner?
    a. Visitor
    b. Iterator
    c. Observer
    d. Builder
24. Care dintre urmatoarele se incadreaza in categoria de Creational Design Patterns:
    a. Singleton
    b. Visitor
    c. Observer
    d. Factory
25. Ce design pattern este folosit pentru a crea obiecte din ierarhia clasei SpaceRock in contextul problemei:
    a. Singleton
    b. Factory
    c. Builder
    d. Strategy
26. Inversion of Control poate fi implementat prin:
    a. Control Pattern
    b. Dependency Injection
    c. Strategy Pattern
    d. Interface Segregation Pattern
27. Ce anti-pattern se refera la un obiect care are prea multe responsabilitati?
    a. Master Object
    b. Spaghetti Code
    c. Procedural Paradigm
    d. God Object
28. Care dintre urmatoarele tehnici poate fi aplicata pentru a pacali o implementare de Singleton bazata pe constructor privat?
    a. Mostenire
    b. Introspectie
    c. Reflectie
    d. Adnotare

PROBLEMA:

La testul practic, studentilor li s-a cerut sa completeze un program, scopul fiind sa arate ca au inteles conceptul de Double Dispatch, cu mentiunea ca orice folosire a operatorului `instanceof` va fi drastic penalizata.

Hacker din fire, unul dintre studenti a produs solutia din codul atasat, liniile scrise de el fiind marcate de comentarii. Haideti sa analizam un pic aceasta solutie:

1. Explicati ce s-ar intampla daca studentul ar apela avoid fara try-catch.
2. Ce ar trebui modificat pentru a implementa corect Double Dispatch? Hint: interfata SpaceRock si clasele care o implementeaza nu ar trebui sa ramana goale, iar codul scris de student ar trebui inlocuit.
3. Comentati cu privire la extensibilitatea programului. Ce fel de modificari ar putea fi dorite si cat de usor sau greu ar putea fi implementate? Comparati, ca extensibilitate, implementarea studentului hacker si cea de la punctul 2.

Programare Orientata pe Obiecte, 7 februarie 2019

1. In Programarea Orientata-Obiect, un obiect este:
    a. o forma geometrica
    b. o lista de instructiuni
    c. o instanta a unei clase
    d. un tip specific de array
2. Care dintre urmatorii sunt specificatori de acces valizi in Java?
    a. void
    b. public
    c. static
    d. protected
3. Metodele interfetei SpaceShip sunt implicit:
    a. public
    b. static
    c. default
    d. private
4. In codul problemei sunt definite:
    a. 7 clase, 2 interfete
    b. 7 clase, 2 interfete, 1 clasa interne anonima
    c. 7 clase, 2 interfete, 2 clase interne anonime
    d. 7 clase, 2 interfete, 3 clase interne anonime
5. Ce design pattern este folosit pentru a crea obiecte din ierarhia clasei SpaceRock in contextul problemei:
    a. Singleton
    b. Factory
    c. Builder
    d. Strategy
6. Suprascrierea este:
    a. Refolosirea unei zone de memorie dupa ce a fost eliberata de Garbage Collector
    b. Redefinirea unei metode intr-o subclasa cu aceiasi parametrii ca in superclasa
    c. Definirea unei metode de mai multe ori intr-o clasa cu liste diferite de parametrii
    d. Alocarea dinamica a unei zone de memorie
7. Adnotarea @Override a metodei `avoid` arata ca:
    a. Metoda este supraincarcata
    b. Metoda trebuie apelata in program
    c. Metoda suprascrie o metoda din superclasa
    d. Metoda trebuie suprascrisa in subclase
8. Care dintre urmatoarele afirmatii este corecta?
    a. Compunerea este o relatie mai puternica decat Agregarea
    b. Agregarea este o relatie de tip IS-A
    c. Mostenirea este o relatie de tip IS-A
    d. Asocierea este o relatie mai puternica decat Agregarea
9. Relatia dintre Asteroid si SpaceRock e de:
    a. Asociere
    b. Agregare
    c. Compunere
    d. Mostenire
10. `(Shuttle)spaceShip` este un exemplu de:
    a. Conversion
    b. Copy prin CopyConstructor
    c. Upcast
    d. Downcast
11. O subclasa aflata in acelasi pachet cu superclasa ei mosteneste:
    a. toate campurile si metodele default
    b. toate campurile si metodele public
    c. toate campurile si metodele private
    d. toate campurile si metodele protected
12. Cand se apeleaza `new Shuttle()` se va invoca:
    a. Un constructor fara parametrii din clasa Shuttle, care nu exista, deci avem eroare
    b. Un constructor fara parametrii declarat implicit in clasa Shuttle
    c. Un constructor fara parametrii pe care clasa Shuttle il mosteneste de la interfata SpaceShip
    d. Un constructor fara parametrii pe care clasa Shuttle il mosteneste de la clasa Solution
13. O clasa obisnuita poate extinde o clasa abstracta:
    a. Daca ii implementeaza toate metodele abstracte
    b. Daca nu are nicio metoda statica
    c. Daca sunt in acelasi pachet
    d. Daca suprascrie cel putin o metoda a clasei abstracte.
14. Daca am dori sa adaugam clasa SpaceXShuttle, aceasta ar avea sens:
    a. Sa fie compusa in clasa Shuttle
    b. Sa extinda clasa Rocket
    c. Sa fie compusa in clasa Rocket
    d. Sa extinda clasa Shuttle
15. Problema diamant apare cand doua clase B si C mostenesc o clasa A, si o alta clasa D mosteneste atat B, cat si C, intrucat:
    a. Daca D suprascrie o metoda din B, pe care C nu o are, rezulta un comportament nedeterminat
    b. Daca B si C trebuie sa suprascrie o metoda din A in acelasi fel se duplica cod
    c. Daca o metoda in A e suprascrisa de B si C, fara ca D sa o suprascrie, este ambiguu pe care implementare o va mosteni
    d. Nu putem face ca C si D sa suprascrie o metoda din A, dar B nu.
16. Ce obiecte vor fi create pentru urmatorul input:

```
shuttle
Asteroid
```

a. un obiect Shuttle si unul Asteroid
b. un obiect Shuttle si unul Meteor
c. un obiect Rocket si unul Asteroid
d. un obiect Rocket si unul Meteor

17. La curs am discutat despre polimorfism:
    a. Static
    b. Dinamic
    c. Parametric
    d. Complex
18. Faptul ca in cod se apeleaza avoid atat cu parametrii de tip Asteroid, cat si Meteor este o folosire a polimorfismului:
    a. Static
    b. Dinamic
    c. Parametric
    d. Complex
19. O clasa interna anonima:
    a. Trebuie sa extinda java.lang.Anonymous
    b. Poate sa extinda o singura clasa sau sa implementeze o singura interfata
    c. Nu are constructori proprii
    d. Este instantiata intr-un singur loc
20. Care dintre urmatoarele este superclasa tuturor erorilor si exceptiilor din Java?
    a. RunTimeException
    b. Catchable
    c. Exception
    d. Throwable
21. Care principiu de design orientat obiect militeaza ca o clasa ar trebui sa aiba un singur motiv pentru care poate fi schimbata?
    a. LSP
    b. SRP
    c. ISP
    d. DIP
22. LSP este reprezentat in codul de la problema prin:
    a. Se foloseste sintaxa try-catch pentru a asigura apelarea corecta a metodei avoid
    b. In main se poate apela avoid pentru obiectul SpaceShip indiferent de implementare, Rocket sau Shuttle
    c. Metoda create este statica
    d. Metoda avoid este supraincarcata
23. ISP este util intrucat, atunci cand este respectat:
    a. implementarile pot fi inlocuite fara a afecta codul care le foloseste
    b. obiectele dintr-un program pot fi inlocuite cu implementari ale subtipurilor
    c. o clasa nu este fortata sa implementeze metode care nu vor fi folosite
    d. reduce numarul de clase si interfete din program
24. Ce design pattern este folosit de clasa Scanner?
    a. Iterator
    b. Observer
    c. Visitor
    d. Builder
25. Care dintre urmatoarele se incadreaza in categoria de Creational Design Patterns:
    a. Visitor
    b. Observer
    c. Singleton
    d. Factory
26. Ce anti-pattern se refera la un obiect care are prea multe responsabilitati?
    a. God Object
    b. Master Object
    c. Spaghetti Code
    d. Procedural Paradigm
27. Inversion of Control poate fi implementat prin:
    a. Control Pattern
    b. Interface Segregation Pattern
    c. Dependency Injection
    d. Strategy Pattern
28. Care dintre urmatoarele tehnici poate fi aplicata pentru a pacali o implementare de Singleton bazata pe constructor privat?
    a. Mostenire
    b. Reflectie
    c. Introspectie
    d. Adnotare

PROBLEMA:

La testul practic, studentilor li s-a cerut sa completeze un program, scopul fiind sa arate ca au inteles conceptul de Double Dispatch, cu mentiunea ca orice folosire a operatorului `instanceof` va fi drastic penalizata.

Hacker din fire, unul dintre studenti a produs solutia din codul atasat, liniile scrise de el fiind marcate de comentarii. Haideti sa analizam un pic aceasta solutie:

1. Explicati ce s-ar intampla daca studentul ar apela avoid fara try-catch.
2. Ce ar trebui modificat pentru a implementa corect Double Dispatch? Hint: interfata SpaceRock si clasele care o implementeaza nu ar trebui sa ramana goale, iar codul scris de student ar trebui inlocuit.
3. Comentati cu privire la extensibilitatea programului. Ce fel de modificari ar putea fi dorite si cat de usor sau greu ar putea fi implementate? Comparati, ca extensibilitate, implementarea studentului hacker si cea de la punctul 2.

```
import java.io.*;
import java.util.*;

interface SpaceRock {}

class Asteroid implements SpaceRock {}

class Meteor implements SpaceRock {}

interface SpaceShip {

    void avoid(Asteroid asteroid);
    void avoid(Meteor meteor);

}

class Shuttle implements SpaceShip {

    @Override
    public void avoid(Asteroid asteroid) {
        System.out.println("Shuttle avoided an Asteroid");
    }

    @Override
    public void avoid(Meteor meteor) {
        System.out.println("Shuttle avoided a Meteor");
    }

}

class Rocket implements SpaceShip {

    @Override
    public void avoid(Asteroid asteroid) {
        System.out.println("Rocket avoided an Asteroid");
    }

    @Override
    public void avoid(Meteor meteor) {
        System.out.println("Rocket avoided a Meteor");
    }

}
```

```
class SpaceShipFactory {

    public static SpaceShip create(String input) {
        if (input.compareTo("shuttle")==0)
            return new Shuttle();
        else
            return new Rocket();
    }

}

class SpaceRockFactory {

    public static SpaceRock create(String input) {
        if (input.compareTo("asteroid")==0)
            return new Asteroid();
        else
            return new Meteor();
    }

}

public class Solution {

    public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        SpaceShip spaceShip =
            SpaceShipFactory.create(sc.nextLine());
        SpaceRock spaceRock =
            SpaceRockFactory.create(sc.nextLine());

        /******** inceput solutie student ********/
        try {
            spaceShip.avoid((Asteroid)spaceRock);
        } catch(Exception e) {

        }

        try {
            spaceShip.avoid((Meteor)spaceRock);
        } catch(Exception e) {

        }
        /******** sfarsit solutie student ********/
    }

}
```