עיצוב מונחה עצמים

מבחן סיום - מועד א' תשס"ב

- יש לענות על 20 מתוך 22 השאלות. רק 20 התשובות הראשונות יבדקו.
- יש לסמן בדיוק תשובה אחת לכל שאלה. סימון לא ברור או כפול ייפסל.
- משקל כל שאלה הוא 5 נקודות.
- השאלות מתייחסות לחומר הלימוד כפי שנלמד בכיתה.
- משך המבחן שעתיים וחצי.
- אין להשתמש בחומר עזר.

בהצלחה! מספר ת.ז.: _____

1. אלו מן הבאים הוא חסרון מרכזי של תבנית העיצוב Visitor?
   א. לא ניתן להשתמש ב-Iterator על מנת לשנות את הסדר בו ה-Visitor עובר על מבנה הנתונים.
   ב. יש לשנות את ממשק ה-Visitor הבסיסי, ולהדר (compile) את כל היורשים שלו, אם מוסיפים מחלקה המגדירה סוג מידע חדש במבנה הנתונים עליו ה-Visitor פועל.
   ג. כל מחלקה המגדירה Visitor חדש (כלומר, מממשת את Visitor ומגדירה אלגוריתם ספציפי) מחייבת הוספת מתודה לקוד מבנה הנתונים עליו ה-Visitor פועל.
   ד. אם משנים מחלקה במבנה הנתונים עלו ה-Visitor פועל כך שהיא תעבוד ע"י Flyweight, יש לשנות את ממשק ה-Visitor הבסיסי.
   ה. לא ניתן לממש Visitors ספציפיים (יורשים מהממשק הבסיסי ומגדירים אלגוריתם ספציפי) בתור Decorators של Visitors אחרים.

2. איזה מהקשרים הבאים ב-UML Class Diagram לא מחייב data member כלשהו כדי לממש אותו?
   א. Aggregation
   ב. Composition
   ג. Dependency
   ד. Association
   ה. אף תשובה אינה נכונה.

3. אתר אינטרנט מאפשר למשתמש בו להגדיר באתר שאילתה מורכבת – הכוללת מספר קריטריונים ותנאים לוגיים ביניהם – שעוברת אופטימיזציה, נשלחת לשרת על מנת להריץ אותה על מסד נתונים, ומציגה למשתמש תוצאה. באיזו תבנית כדאי להשתמש כדי לבנות את אובייקט השאילתה?
   א. Builder
   ב. Prototype
   ג. Observer
   ד. State
   ה. Abstract Factory

4. מאלו מעקרונות העיצוב הבאים נובע שמחלקה היורשת ממחלקה אחרת חייבת לקיים את ה-Invariant של המחלקה שירשה?
   א. Open-Closed Principle
   ב. Liskov Substitution Principle
   ג. Law of Demeter
   ד. Command-Query Separation
   ה. Single Choice Principle

5. איזו מהתבניות הבאות מתאימה ביותר לשימוש ב-Black-Box Framework?
   א. Abstract Factory
   ב. Builder
   ג. Prototype
   ד. Factory Method
   ה. Command

6. אלו מהטענות הבאות לגבי Creational Design Patterns איננה נכונה?
   א. יש יתרון בהגדרת Builder המשתמש באופן פנימי ב-Abstract Factory, על מנת להרוויח הן יכולת החלפה קלה בין חלקי האובייקט המורכב הנוצר, והן יכולת לאופטימיזציה בסיום בנייתו.
   ב. יש יתרון בהגדרת Builder המשתמש באופן פנימי ב-Prototype, על מנת להרוויח הן החלפה קלה בין חלקי האובייקט המורכב הנוצר, והן יכולת לאופטימיזציה בסיום בנייתו.
   ג. התבנית Prototype מאפשרת גמישות רבה ביותר, כיוון שניתן דרכה לשלוט לא רק על המחלקה של האובייקט הנוצר, אלא גם על אופן האתחול של נתוני האובייקט.
   ד. אחד החסרונות של Abstract Factory הוא הצורך לקודד מחלקה חדשה בכל פעם שרוצים לשנות אפילו אחת מן המחלקות המשמשות ליצירת אובייקטים ב-Factory קיים.
   ה. מבין התבניות Abstract Factory, Builder, Prototype, השימוש ב-Prototype ידרוש את ההוספה של מספר המחלקות הגדול ביותר לתוכנית.

7. אלו מהטענות הבאות איננה נכונה?
   א. מטרת התבנית Strategy היא לאפשר החלפה בזמן ריצה בין מספר אלגוריתמים הפותרים את אותה בעיה.
   ב. מטרת התבנית Visitor היא להפריד בין מבנה נתונים לאלגוריתמים מורכבים הפועלים עליו.
   ג. מטרת התבנית Observer היא להגדיר קשרי יחיד-לרבים בין אובייקטים, כך ששינוי ביחיד יעדכן את כל האובייקטים הקשורים אליו.
   ד. מטרת התבנית Composite היא לשתף אובייקטים על מנת לתמוך באופן יעיל וחסכוני במספר גדול של אובייקטים קטנים, הכוללים data members הנמצאים בשימוש נדיר.
   ה. מטרת התבנית Abstract Factory היא לאפשר החלפה בזמן ריצה של המחלקות שיש ליצור בבקשת יצירה של אובייקטים מרשימת סוגים נתונה.

8. באיזה מהמצבים הבאים לא כדאי להשתמש ב-Refactoring?
   א. על מנת לתקן באג
   ב. אם הפרויקט עומד בפני deadline קרוב
   ג. אם הפרויקט קטן יחסית
   ד. אם יש מעט מאד unit tests
   ה. כל התשובות נכונות

9. איזו מהתבניות הבאות ניתן להוסיף לקוד של מחלקה קיימת בלי לשנות את ה-interface שלה?
   א. State
   ב. Observer
   ג. Visitor
   ד. Iterator
   ה. אף תשובה אינה נכונה.

10. אילו מהטענות הבאות איננה נכונה?
    א. התבנית Prototype צורכת זיכרון רב יחסית לתבניות יצירה אחרות.
    ב. התבנית Template Method אינה מספקת כל גמישות בזמן ריצה.
    ג. התבנית Flyweight עלולה לגרום לבזבוז זיכרון אם מתבצעת חלוקה שגויה בין המידע המשותף והלא-משותף במבנה הנתונים המשתמש ב-Flyweight.
    ד. התבנית Observer ניתנת להרחבה בקלות כדי לאפשר לאובייקט להירשם כמאזין למספר אובייקטים מאותה מחלקה.
    ה. התבנית Mediator מחייבת את כל האובייקטים שיש ביניהם אינטראקציה מסוימת (למשל, שינויים בהם עשויים לגרור שינויים באחרים) להכיר (ע"י קשר association) זה את זה.

11. לאילו מבין זוגות התבניות הבאות יש מימוש זהה בקוד?
    א. Decorator ו-Proxy
    ב. Observer ו-Mediator
    ג. Flyweight ו-Prototype
    ד. Strategy ו-Command
    ה. Builder ו-Visitor

12. מנגנון ה-templates של שפת C++ להגדרת אוספים, למשל list<Person*>, עדיף על המנגנון המקביל של Java המאפשר להגדיר רק List המחזיק ערכים מטיפוס Object כללי, כי:
    א. הוא מאפשר למהדר (compiler) לייצר קוד יעיל יותר
    ב. הוא חוסך פעולות הטלה (casting) בקוד
    ג. הוא מאפשר לבצע בדיקות רבות יותר בזמן הידור במקום בזמן ריצה
    ד. תשובות א ו-ב נכונות
    ה. כל התשובות נכונות

13. אלו מהבאים הוא דוגמא למצב בו לא נכון לזרוק Exception?
    א. בתחילת ריצתה פונקציה מגלה שהפרמטרים שהועברו לא מקיימים את ה-precondition שלה
    ב. בסוף ריצת פונקציה מתגלה שהיא לא מקיימת את ה-postcondition שלה
    ג. כתיבת מידע לקובץ נכשלת בגלל שגיאת חומרה
    ד. נכשלת בקשה להקצאת זיכרון ממערכת ההפעלה
    ה. פונקציה המחפשת האם אובייקט נמצא ברשימה מחזירה את האובייקט אם הוא נמצא בה, או זורקת Exception אם הוא לא

14. המחלקה Stack<T> ממשת מחסנית ע"י מערך data ואינדקס top המציין את מספר האיבר בראש המחסנית. בפונקציה הבאה: T pop() { return data[top--]; }
עלולה להיזרק Exception ב-copy constructor של המחלקה T, ועקב כך פונקציה זו אינה מקיימת:
    א. Exception Neutrality
    ב. Weak Exception Safety
    ג. Strong Exception Safety
    ד. Law of Demeter
    ה. כל התשובות נכונות.

15. אלו מהטענות הבאות לגבי Command-Query Separation אינה נכונה?
    א. יש מצבים בהם לא ניתן להשיג Exception Safety בשפת C++ בלי לקיים עקרון זה.
    ב. ממשקים של DCOM לא צריכים לקיים עקרון זה, מטעמי ביצועים.
    ג. זו שגיאה חמורה להפר עקרון זה בפונקציות שנעשה בהן שימוש בהגדרת assertions.
    ד. הקפדה על עקרון זה במחלקה מסוימת עשויה להגדיל את מספר ה-methods בה.
    ה. עקרון זה יוצר בדרך כלל מחלקות אלגנטיות יותר, אבל הוא מוריד את ביצועי המערכת ומונע אופטימיזציות של המהדר, עקב הצורך לבצע קריאות רבות יותר לפונקציות.

16. מי מהבאים אינו אחד מהעקרונות של Extreme Programming?
    א. Continuous Integration
    ב. Refactoring
    ג. Pair Programming
    ד. Small Releases
    ה. Design by Contract

17. ממשק COM בשם IClock מומש בשפת C++ ע"י המחלקה CClock. מה מהבאים ניתן לבצע בתוכנית בשפת Delphi המשתמשת במימוש זה?
    א. להגדיר מחלקה CAdvancedClock היורשת מהמחלקה CClock ומשנה חלק מהמימושים בה
    ב. לתפוס Exceptions שנזרקו ע"י הקוד בתוך CClock ולטפל בהן בתוך קוד Delphi
    ג. לבקש ממערכת ההפעלה ליצור אובייקט מהמחלקה CClock, ולקרוא לפונקציות עליו
    ד. לבקש ממערכת ההפעלה ליצור אובייקט המממש את הממשק IClock, ולקרוא לפונקציות עליו
    ה. אף תשובה אינה נכונה

18. אלו מהבקשות הבאות לא נתמכת ע"י מנגנון ה-RTTI של שפת C++?
    א. קבלת שם הטיפוס (כ-string) של אובייקט נתון
    ב. בדיקה האם שני אובייקטים הם מאותו טיפוס בדיוק
    ג. יצירת אובייקט בהינתן שם המחלקה שלו כ-string
    ד. בדיקה האם ניתן להשים (ע"י assignment operator) אובייקט נתון במשתנה מטיפוס מסוים
    ה. אף תשובה אינה נכונה

19. איזה מהמקרים הבאים מתאר שימוש שגוי במנגנון ה-RTTI של שפת Java?
א. שימוש ב-instanceof כדי לבדוק את הטיפוס המדויק של איבר במבנה נתונים פולימורפי, על מנת לבצע פעולה רק על איברים מטיפוס זה
ב. שימוש ב-casting על אובייקט שהגיע מהרשת כדי לוודא את הטיפוס שלו
ג. שימוש ב-instanceof כדי לבחור אחד מבין כמה מימושים אפשריים (ע"י if…else if) ע"פ טיפוס האובייקט המשתתף במימוש, כאשר אי אפשר לשנות את הקוד של המחלקה של אובייקט זה
ד. שימוש ב-casting של אובייקט מן המחלקה של המשתנה המצביע אליו למחלקה היורשת ממנה, כדי לקרוא לפונקציות המוגדרות במחלקה היורשת בלבד
ה. שימוש ב-casting כדי לתרגם את האובייקט החוזר מיצירת dynamic proxy instance לאחד הממשקים שהוא תומך בהם

20. איזו מהטענות הבאות לגבי מנגנון ה-Dynamic Proxies של שפת Java נכונה?
א. אם הפונקציה (invoke קוראת בין השאר למימוש המקורי של האובייקט הנעטף, אז פונקציה זו לא יכולה להיות Exception Neutral, כיוון שלא ניתן להבדיל בין שגיאות שעלו ממנה לשגיאות מאותו טיפוס שעלו מן המימוש הנעטף.
ב. מחלקה שהיא dynamic proxy class יכולה לממש כמה ממשקים (interfaces), וגם להחליף את הממשקים שהיא תומכת בהם לאחר יצירתה.
ג. יצירת dynamic proxy instance המממש ממשק X איטית יותר מיצירת אובייקט Java ממחלקה רגילה (המוגדרת בקוד) המממשת ממשק זה, אולם לאחר יצירת האובייקט קריאה לפונקציה עליו אינה איטית יותר מקריאה לאותה פונקציה על אובייקט רגיל
ד. כל יכולת שניתן לכתוב בעזרת dynamic proxies ניתן לכתוב גם בעזרת תבנית העיצוב Proxy, כך שכמות הקוד שיש לכתוב תישאר דומה, אך המימוש יהיה פחות יעיל מבחינת זמן ריצה וצריכת זיכרון
ה. אף תשובה אינה נכונה

21. אפליקציה לעריכת יצירות מוזיקליות משתמשת במבנה נתונים עיקרי במחלקה MusicItem, ועל פי התבנית Composite מגדירה למחלקה זו יורשים פשוטים (תו בודד או סימן מוזיקלי) ויורשים מורכבים (שורת תווים, דף תווים וכדומה) שכולם יורשים מ-MusicItem. על אפליקציה זו לממש יכולת Cut / Copy / Paste כללית – כלומר, באפליקציה ניתן לבחור כל MusicItem ואז לגזור ולהדביק אותו. כדי לממש העתקה, הוגדרה במחלקה MusicItem פונקציה בשם (clone המחזירה עותק של האובייקט המועתק. אלו מהטענות הבאות אינה נכונה?
א. המימוש של (clone במחלקה CompositeMusicItem ייצור אובייקט חדש ריק מסוגו, וימלא אותו בעותקים של האובייקטים המוכלים בו, שנוצרו ע"י קריאת clone רקורסיבית עליהם.
ב. כדי לממש גזירה (Cut) בצורה יעילה, על כל אובייקט במבנה הנתונים להכיר את האובייקט בו הוא מוכל, כלומר כדאי להוסיף למחלקה MusicItem את הפונקציה (getParent.
ג. ניתן לייעל משמעותית את זמן הריצה וצריכת הזיכרון של פעולות Copy על אובייקטים גדולים על ידי יצירת Proxy המממש את אופטימיזצית Copy-on-write במקום ביצוע ההעתקה עצמה.
ד. מימוש מנגנון גזירה והדבקה ע"י (clone לא אפשרי אם חלק מהמחלקות היורשות של MusicItem משתמשות בתבנית Flyweight.
ה. ניתן לבצע פעולת Copy גם בזמן שקיימים איטרטורים (Iterators) פעילים על מבנה הנתונים. לעומת זאת, סוגי איטרטורים מסוימים לא יעבדו כנדרש לאחר ביצוע פעולת Cut בזמן פעולתם.

22. אלו מהבאים הוא דוגמא לסגנון תכנות שגוי ע"פ עקרונות Design by Contract?
א. בדיקת ה-precondition של פונקציה בתחילת המימוש שלה ע"י תנאי if..else
ב. בקשה מה-compiler לא לבצע בדיקות שחוזים (assertions) אכן מתקיימים בזמן ריצה, כדי לחסוך את השפעת בדיקות אלו על ביצועי התוכנית
ג. הכללת החוזה של מחלקה כחלק מן התיעוד שלה (למשל כחלק מהפלט של JavaDoc)
ד. בשפת Java, הגדרת המחלקה AssertionViolationException, הנזרקת אם מתגלה בזמן ריצה הפרה של חוזה, כיורשת של RuntimeException, וכך הימנעות מהכללתה ברשימת ה-Exceptions שהפונקציה המכילה את החוזה עלולה לזרוק
ה. כל התשובות נכונות