עיצוב מונחה עצמים

מבחן סיום - מועד א' תשס"ד

- 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 מהטענות הבאות לגבי UML Diagrams אינה נכונה?
   א. Use Case Diagram היא תיאור של תהליך עבודה אחד מלא עם המערכת
   ב. Class Diagram מתארת מבנה סטטי של הקשרים בין המחלקות השונות במערכת
   ג. Sequence Diagram מתארת תהליך דינאמי של קריאות בין אובייקטים
   ד. Collaboration Diagram היא דרך נוספת להציג את אותו מידע המוצג ע"י Sequence Diagram
   ה. ניתן ב-Sequence Diagram להביע תנאים ולולאות

2. אלו מהפעולות הבאות לא ניתן לבצע בשפה כמו C++, שתומכת ב-RTTI אבל לא ב-Reflection?
   א. בדיקה אם אובייקט יורש ממחלקה מסוימת
   ב. הדפסת שם המחלקה של אובייקט כ-string
   ג. יצירת אובייקט ממחלקה מסוימת בהינתן שם המחלקה כ-string
   ד. בדיקה האם שני אובייקטים הם מאותו טיפוס
   ה. המרה (casting) של מצביע לממשק למצביע למחלקה קונקרטית המממשת אותו

3. אלו מהשגיאות הבאות שמתגלות בדרך כלל בזמן קומפילציה, אי אפשר ליצור בזמן ריצה ע"י שימוש שגוי ב-Reflection?
   א. קריאה למתודה תוך העברת מספר שגוי של פרמטרים
   ב. קריאה למתודה שאינה קיימת במחלקה עליה מפעילים אותה
   ג. החזרת ערך ממתודה שאינו מטיפוס ההחזר שהוגדר לה
   ד. העלאת Exception מתוך מתודה שלא הוגדר ברשימת ה-throws שלה
   ה. אף תשובה לא נכונה

4. אלו מהדברים הבאים לא ניתן לעשות בסביבת NET. בקריאה מקוד שכתוב בשפה אחת לקוד שכתוב בשפה אחרת?
   א. להעלות Exception, שייתפס במנגנון ה-Exceptions של השפה השנייה
   ב. לרשת ממחלקה שנכתבה בשפה השנייה, ולשנות מימוש של פונקציה וירטואלית
   ג. ליצור אובייקט ממחלקה שנכתבה בשפה השנייה
   ד. לקרוא למתודות של אובייקט COM שנכתב בשפה השנייה
   ה. לקרוא לקוד שנכתב כ-unmanaged (unsafe) code ולא קומפל ל-IL בשפה השנייה

5. תוכנית Java מימשה וריאציה של התבנית Command, בה כל Object (ולא רק יורשים של הממשק Command) יכולים להגדיר פקודה להרצה ע"י מימוש מתודה בשם execute, ובהרצת פקודה מתפריט מורצת מתודה זו באובייקט המקושר לפקודה ע"י reflection. אלו מהטענות הבאות לגבי מנגנון זה נכונה?
   א. הוא מפעיל פקודות באיטיות רבה יותר מן המימוש הרגיל של התבנית Command
   ב. הוא בהכרח צורך זיכרון רב יותר מאשר המימוש הרגיל של Command
   ג. ניתן לבדוק בזמן קומפילציה שכל אובייקט המקושר לפקודה כולל מתודה בשם execute
   ד. לא ניתן להרחיב אותו לביצוע undo (ע"י הגדרת כלל שאובייקטים המגדירים פקודות ניתנות לביטול יממשו מתודה נוספת בשם undo)
   ה. לא ניתן לממש בו פעם אחת CompositeCommand שמכיל סדרה של פקודות אחרות ויודע לבצע אותן סדרתית

6. למה טכנולוגיית Web Services מקלה את האינטגרציה בין מערכות מידע הקשורות זו לזו?
   א. כי בניגוד לטכנולוגיות קודמות, היא משתמשת בסטנדרטים פתוחים שמומשו גם בסביבות Windows וגם בסביבות Unix
   ב. כי שתי מערכות מקושרות צריכות להסכים ביניהן רק על הסכמה שעוברת ברשת, ולא צריכות להחזיק קוד משותף זהה בשני קצות הקשר לפענוח המידע העובר
   ג. כי מספיק שאחת המערכות תדע לקבל בקשות בסטנדרט HTTP על מנת להעביר מידע
   ד. כי פורמט ההודעות העובר בין מערכות הוא XML טקסטואלי, ולכן אין צורך להתמודד עם בעיות כגון ייצוג שונה של מידע בינארי בין חומרות שונות וכדומה
   ה. כל התשובות נכונות

7. באלו מן המקרים הבאים לא ניתן להשתמש בתבנית Decorator?
   א. על גבי התבנית Command, כדי לאפשר למשל להגדיר פקודה חדשה, המבצעת פקודה אחרת כלשהי, אבל רק כאשר תנאי מסוים מתקיים
   ב. על גבי התבנית Iterator, כדי לאפשר למשל להגדיר איטרטור חדש, העובר רק על כל איבר שני של איטרטור אחר כלשהו
   ג. על גבי התבנית Composite, כדי לאפשר למשל להגדיר איבר חדש בעץ ש-Composite מגדיר, שיכול להוסיף לכל איבר אחר בעץ יכולת נוספת כלשהי
   ד. על גבי התבנית Strategy, כדי לאפשר למשל להגדיר אסטרטגיה חדשה, שיכולה להריץ כל אסטרטגיה אחרת אך מוסיפה כתיבה לקובץ לוג של התוצאה שאותה אסטרטגיה החזירה
   ה. על גבי התבנית Template Method, כדי לאפשר למשל לעטוף את המתודה שנכתבה לצורך התבנית במתודה אחרת, שמפעילה את המתודה המקורית רק אם מתקיים תנאי מסוים

8. אלו מהדברים הבאים לא צריך מהדר (compiler) לשפה מסוימת שתומך ב-COM לדעת לבצע?
   א. להחליף יצירת אובייקטי COM ע"י אופרטור היצירה של השפה (למשל new) ליצירתם ע"י קריאה לפונקציה של מערכת ההפעלה
   ב. לתרגם לשפת מכונה קריאות לפונקציות על אובייקטי COM לפי ה-calling convention של COM, ולאו דווקא של אותה שפה
   ג. לתרגם לשפת מכונה מימוש של אובייקטי COM חדשים כך שהפעלה של פונקציות אליהם תהיה לפי הסטנדרט של virtual tables, ולאו דווקא מימוש קריאות לפונקציות בשפה
   ד. לדעת לתרגם כל טיפוס (type) באותה שפה המועבר למתודת COM לטיפוס המוכר ע"י COM
   ה. אף תשובה לא נכונה – כלומר, המהדר חייב לדעת לעשות את כל ארבעת הסעיפים הנ"ל

9. אלו מהיכולות הבאות של Aspect-Oriented Programming לא ניתן לממש בשפת Java ע"י Dynamic Proxies? הניחו שיצירת אובייקטים נעשית רק דרך ה-Proxy.
   א. להוסיף קטע קוד שמורץ לפני קריאה לכל מתודה של ממשק מסוים במחלקה המממשת אותו, בלי להוסיף את הקוד לאותה מחלקה
   ב. להוסיף למחלקה ממשק חדש שהיא מממשת, וכן מימוש למתודות שלו, בלי לשנות את הקוד של אותה מחלקה
   ג. לתפוס את כל ה-Exceptions מסוג מסוים שמחלקה זורקת מתוך המתודות של ממשק שהיא מממשת, ולהחליפם ב-Exception אחר, בלי לשנות את הקוד של המחלקה
   ד. לכתוב לקובץ לוג מרכזי את מספר הפעמים הכולל שמתודה מסוימת של ממשק נקראה במהלך ריצת התוכנית, מכל האובייקטים של מחלקות שמימשו ממשק זה
   ה. להוסיף קטע קוד שמורץ לפני עדכון של public data field של מחלקה מסוימת, בלי להוסיף את הקוד לאותה מחלקה

10. מה ההבדל בין static typing לבין static binding?
    א. Static binding דורש שלכל ביטוי בשפה יהיה טיפוס שמוגדר בזמן קומפילציה, בעוד ש-static typing דורש שכל קריאה למתודה תקושר למימוש יחיד שמוגדר בזמן קומפילציה
    ב. Static typing דורש שלכל ביטוי בשפה יהיה טיפוס שמוגדר בזמן קומפילציה, בעוד ש-static binding דורש שכל קריאה למתודה תקושר למימוש יחיד שמוגדר בזמן קומפילציה
    ג. שפה התומכת ב-static binding ולא ב-dynamic binding יכולה להיות שפה מונחת עצמים, אבל שפה התומכת ב-static typing ולא ב-dynamic typing לא יכולה
    ד. תמיכה ב-Static Binding מחייבת גם תמיכה ב-Static Typing, אבל לא להיפך
    ה. תמיכה ב-Static Typing מחייבת גם תמיכה ב-Static Binding, אבל לא להיפך

11. מאיזה עקרון עיצוב נובע כי כל המתודות של מחלקה חייבות לקיים את ה-Post-conditions של אותן המתודות במחלקה ממנה היא יורשת?
    א. Open-Closed Principle
    ב. Liskov Substitution Principle
    ג. Law of Demeter
    ד. Command-Query Separation
    ה. Single Choice Principle

12. נניח שממנגנון ה-Reflection של שפת Java הוסרה האפשרות להפעיל מתודות, כלומר נמחקה המתודה ()invoke של המחלקה Method. אלו מהדברים הבאים לא יהיה ניתן לממש יותר:
    א. הדפסה של שמות כל השדות והמתודות של אובייקט נתון
    ב. הדפסת שם המחלקה של אובייקט, וכל אבותיו בעץ הירושה
    ג. מימוש מנגנון גנרי דמוי serialization, שמקבל אובייקט מכל מחלקה שהיא, ויודע לשמור אותו לפורמט בינארי, ולטעון אובייקט חדש מאותה מחלקה על סמך פורמט זה
    ד. מימוש מנגנון plugins, שקורא בזמן ריצה רשימה של plugins קיימים, טוען ממיקום מוסכם מראש את המחלקות המממשות אותן, ומשתמש באובייקטים מאותן מחלקות
    ה. מימוש מנגנון trace לצורכי debug בעזרת dynamic proxy, כלומר מנגנון שמאפשר שתילת קוד של כתיבה לקובץ לוג, לפני ואחרי כל קריאה למתודה של כל מחלקה שהיא

13. אלו מהטענות הבאות לגבי MVC אינה נכונה?
    א. אובייקט ה-Model הוא בעיקרו מבנה נתונים, שמחזיק רשימת Observers
    ב. אובייקט ה-View צריך להכיר את ה-Model, אבל לא את ה-Controller
    ג. אובייקט ה-Controller צריך להכיר את ה-Model אבל לא את ה-View
    ד. ניתן בקלות לתמוך בכמה Views שונים על אותו מודל, שמספרם מוגדר בזמן ריצה
    ה. שימוש בתבנית MVC מחייב את מימוש שלושת חלקיה כשלושה אובייקטים שונים

14. אלו מהטענות הבאות לגבי תבנית העיצוב Composite נכונה?
    א. ניתן לממש את תבנית העיצוב Composite בשתי דרכים: המחלקה CompositeShape יכולה לרשת ירושה מרובה מ-Shape וגם מ-list<Shape*>, או לרשת ירושה בודדת מ-list<Shape*> ולהחזיק data member שהוא מצביע ל-Shape
    ב. ניתן לממש את תבנית העיצוב Composite בשתי דרכים – ע"י ירושה מרובה או ע"י ירושה בודדת והצבעה. המימוש של ירושה מרובה דורש כתיבת יותר קוד.
    ג. ניתן לבחור בכל מבנה נתונים לאחזקת רשימת הילדים של כל Composite, אך יש להשתמש אך ורק במבנה נתונים יחיד זה לאחזקת כל האובייקטים במבנה הנתונים.
    ד. במידה ולמערכת קיימת מוסיפים מחלקה חדשה היורשת מ-Shape, יש לשנות את הקוד של CompositeShape רק במקום אחד על מנת להכניס אליו גם אובייקטים מהסוג החדש
    ה. אף אחת מהטענות הנ"ל אינה נכונה

15. אלו מהבאים הוא ההבדל בין התבניות Strategy ו-State?
    א. בניגוד ל-State, אובייקט Strategy לא יכול להגדיר משתנים (data members) משל עצמו
    ב. בניגוד ל-Strategy, ניתן להחזיר רק אובייקט State יחיד קיים בזיכרון עבור כל אובייקט, וכל מעבר בין מצבים מחייב את מחיקת אובייקט המצב הקודם, ויצירת אובייקט עבור המצב החדש
    ג. בניגוד לאובייקט Strategy, לאובייקט State אסור לשנות את האובייקט המכיל אותו (למשל, אסור לו לגרום להחלפת ה-State הנוכחי של אותו אובייקט)
    ד. בתבנית Strategy יש לספק public methods באובייקט העוטף על מנת לאפשר החלפת אסטרטגיה, ולעומת זאת אסור לספק מתודות כאלה בתבנית State
    ה. אף תשובה אינה נכונה

16. מה מהבאים הוא יתרון של COM בפיתוח אפליקציות, שאינו קיים בפיתוח מונחה עצמים רגיל?
    א. ממשק בינארי המאפשר שילוב קל של רכיבים שנכתבו בשפות שונות
    ב. אפשרות לשאול בזמן ריצה אלו ממשקים רכיב נתון מממש
    ג. תמיכה מובנית בהעלאת Exceptions בין שפות תכנות שונות
    ד. תמיכה בירושה בין ממשקים (COM Interfaces)
    ה. ניהול זיכרון ע"י reference counting

17. תוכנית צריכה לדעת לבנות מבנה נתונים המתאר מפה של עיר, אך סוגי הבניינים וסוגי הכבישים בהם יש להשתמש בבניית המפה עשויים להשתנות. לכל סוג בניין וסוג כביש יש מחלקה המתארת אותו, היורשת מ-Building או מ-Road בהתאמה. איזו מתבניות העיצוב הבאות תדרוש הוספה של מספר המחלקות הקטן ביותר כדי לממש בקוד שינוי של סוג בניין?
    א. Prototype
    ב. שילוב של Builder ו-Abstract Factory
    ג. Factory Method
    ד. Abstract Factory
    ה. Builder

18. עבור התוכנית מן השאלה הקודמת, מהו הפתרון היעיל ביותר מבחינת זמן ריצה?
    א. Prototype
    ב. שילוב של Builder ו-Abstract Factory
    ג. Factory Method
    ד. Abstract Factory
    ה. Builder

19. אלו מהתבניות הבאות הן מימוש אפשרי ל-Hook של Black-Box Framework?
    א. Visitor
    ב. Prototype
    ג. Strategy
    ד. Observer
    ה. כל התשובות נכונות

20. אלו מהטענות הבאות לגבי תכנות גנרי (Generic Programming) היא נכונה?
    א. תכנות גנרי מאפשר לכתוב container עבור טיפוס מסוים, כך שאותו טיפוס נקבע בזמן ריצה
    ב. תכנות גנרי מאפשר לכתוב container עבור טיפוס מסוים, כך שאותו טיפוס נקבע בזמן הידור
    ג. אם יש שני טיפוסים Foo ו-Bar כך ש-Foo יורש מ-Bar, אז באופן אוטומטי vector<Foo> יורש גם כן מ-vector<Bar>
    ד. אם Foo ו-Bar הן שתי מחלקות בשפת C++, אז vector<Foo*> ו-vector<Bar*> בהכרח יגרמו למהדר ליצור קוד נפרד עבור שני המימושים השונים של vector<>
    ה. כל התשובות נכונות

21. אלו מהטענות הבאות נכונה?
    א. משמעות Weak Exception Safety היא שלאחר זריקת Exception, מובטח שה-state של האובייקט אינו משתנה
    ב. משמעות Strong Exception Safety היא שלאחר זריקת Exception, מובטח שה-state של האובייקט אינו משתנה
    ג. בשפת C++, אין בעיה לזרוק Exception מתוך Constructor, כי הקצאת הזיכרון עבורו תתבטל
    ד. משיקולי יעילות עדיף לזרוק ולתפוס Exception, מאשר לבדוק אם הפעולה עומדת להיכשל (למשל באמצעות if)
    ה. בשפת C++ לא ניתן לזרוק Exception ב- Copy Constructor

22. בשפת Java קיים ממשק בשם java.util.Iterator המאפשר לעבור על Collection כלשהו. את ממשק זה מרחיב (extends) הממשק java.util.ListIterator שמיועד לעבור על רשימות בלבד. לממשק זה (ListIterator) יש (בין השאר) את המתודה previous() שמחזירה את האובייקט הקודם לזה שה- ListIterator מצביע עליו, ובו בעת משנה את ההצבעה של ה- ListIterator לאיבר הקודם ברשימה. אלו מעקרונות העיצוב הבאים מפרה המתודה previous()?
    א. Open Closed Principle
    ב. Dependency Inversion Principle
    ג. Interface Segregation Principle
    ד. Acyclic Dependencies Principle
    ה. Command Query Separation