תעודת זהות:

|  |  |
| --- | --- |
| שנה"ל: | תשע"ח |
| סמסטר קיץ | מועד ב |
| תאריך: | 18/10/2018 |
| יום: | שני |
| שעה: | 16:00 |
| משך הבחינה: | 3 שעות |
| חומר עזר מותר: | חומר פתוח ומחשבון |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_ |

##### בחינה בקורס: תכנות מוכוון עצמים ושפת JAVA

## שם מרצה: מר פרופוס אפי

**מדבקית**

**ברקוד**

**הנחיות:**

1. המבחן מכיל 3 שאלות. יש לענות על כולן. הציון המכסימלי הוא 100.
2. יש לענות על כל השאלות במחברת הבחינה. יש להחזיר את השאלון.
3. הערות:
   1. שימו לב לכתוב קוד פשוט וברור!
   2. יש לכתוב קוד אופטימלי שקל להתמצא בו ולבצע תוספות ושינויים
   3. יש להקפיד על שמות משמעותיים!
   4. יש לתעד כל מה שאינו טריוויאלי!
   5. ניתן לכתוב פונקציות עזר במידת הצורך כדי לשמור על מודולריות!

**טופס הבחינה כולל 8 עמודים (כולל דף זה )**

**ב ה צ ל ח ה**

**שאלה 1 ( 30 נקודות ):**

1. ( 15 נקודות ) מה יודפס כאשר נריץ את התוכנית הבאה:

**class** Exc0 **extends** Exception { }

**class** Exc1 **extends** Exc0 { }

**class** Exc2 **extends** Exc1 { }

**public** **class** main

{

**public** **static** **void** main(String args[])

{

**try**

{

**throw** **new** Exc2();

}

**catch** (Exc1 e1)

{

System.***out***.println("caught me ");

}

**catch** (Exception e)

{

System.***out***.println("catch me if you can");

}

}

}

1. ( 15 נקודות )

נתונות הגדרות למחלקות של: סטודנט (Student), סטודנט לתואר ראשון (BAStudent), סטודנט לתואר שני (MasterStudet), וסטודנט לתואר שלישי (PhDStudent):

**public** **abstract** **class** Student {

**private** String institute;

**private** **int** year;

**private** **int** id;

**protected** String name;

**public** Student(**int** id, String name, String institute, **int** year){

**this**.id = id;

**this**.name = name;

**this**.institute = institute;

**this**.year = year;

}

**public** **int** whichYear(){**return** year;}

**public** String whichInstitute(){**return** institute;}

**public** **int** getId(){**return** id;}

**public** String getName(){**return** name;}

**public** String whoIs(){

**return** getClassName()+" "+getName()+" id = "+getId()+

" from "+whichInstitute()+", "+whichYear()+" year";

}

**abstract** **void** introduce();

**abstract** String getClassName();

}

**public** **class** BAStudent **extends** Student {

**private** **double** average;

**public** BAStudent(**int** id, String name, String institute,

**int** year, **double** avg)

{

**super**(id, name, institute, year);

**this**.average = avg;

}

**public** **double** getAverage(){**return** average;}

**public** String whoIs(){

**return** **super**.whoIs()+" has an average grade "+getAverage();

}

**public** String getClassName() {**return** "B.Sc. Student";}

**void** introduce() {

System.*out*.println("Greetings from "+name);

}

}

**public** **class** MasterStudent **extends** Student {

**private** String thesis;

**private** String advisor;

**private** String title = "Bachelor of Science ";

**public** MasterStudent(**int** id, String name, String inst, **int** year,

String thesis, String advisor) {

**super**(id, name, inst, year);

**this**.name = title+name;

**this**.thesis = thesis;

**this**.advisor = advisor;

}

**public** String whoIs(){

**return** **super**.whoIs()+" works with "+getAdvisor();

}

**public** String getThesis() {**return** thesis;}

**public** String getAdvisor() {**return** advisor;}

**public** **void** research(){

System.*out*.println(getClassName()+" named "+getName()+

" makes research in "+getThesis()+

" under supervision of "+getAdvisor());

}

**public** String getClassName() {**return** "M.Sc. Student";}

**void** introduce() {

System.*out*.println("Best wishes from "+name);

}

}

**public** **class** PhDStudent **extends** MasterStudent {

**public** PhDStudent(**int** id, String name, String inst, **int** year,

String thesis, String advisor) {

**super**(id, "Master of Science "+name, inst, year, thesis,

advisor);

}

**public** String whoIs(){

**return** **super**.whoIs()+" on dissertation "+getThesis();

}

**public** String getClassName() {

**return** "Ph.D. Student";

}

**public** String getName(){**return** name;}

**void** introduce() {

System.*out*.println("With respect from "+getName());

}

}

1. כתוב פלט (7 שורות, יש למספר מ-1 עד-7) עבור קטע קוד הבא:

**public** **static** **void** main(String[] args) {

Student s1 = **new** BAStudent(123, "Moshe", "Academic", 2, 80);

s1.introduce();

System.*out*.println(s1.whoIs());

Student s2 = **new** MasterStudent(123, "Peter", "BGU", 1, "What what?",

"Prof. Who Who?");

s2.introduce();

System.*out*.println(s2.whoIs());

Student s3 = **new** PhDStudent(123, "Samir", "MIT", 2, "XYZ",

"Prof. X Y");

s3.introduce();

((PhDStudent)s3).research();

System.*out*.println(s3.whoIs());

}

**שאלה 2 ( 40 נקודות ):**

חלק א ( 30 נקודות ):

עליכם להגדיר את המחלקות הבאות:

* **סטודנט** (Student).
* **עובד** (Employee).
* **עובד משרת סטודנט** (StudentEmployee).

לכל בנאדם (סטודנט ועובד) מוגדר:

* **שם** (String name),
* **מס' ת"ז** (String id)

בנוסף, לכל סטודנט מוגדר:

* **מחלקה** שבה הוא לומד (String dept),
* **שנת לימודים** (int year)

כמו כן, לכל עובד מוגדר:

* **מקום עבודה** שבו הוא עובד (String workplace),
* **משרה** (String job)

לכל בנאדם יש פונקציה **public** **void** introduce() המדפיסה פרטים הבאים:

* **שם** **ומס' ת"ז** עבור כל אחד.
* **מחלקה** **ושנת לימודים** עבור סטודנט (בנוסף לשם ות"ז)
* **מקום עבודה** ו**משרה** עבור עובד (בנוסף לשם ות"ז)
* **כל הפריטים הנ"ל** עבור עובד משרת סטודנט

לכל סטודנט יש פונקציה **public** String studentInfo() המחזירה מחרוזת המכילה ייצוג טקסטואלי של אותו הסטודנט: מחלקה ושנת לימודים.

לכל עובד יש פונקציה **public** String workInfo() המחזירה מחרוזת המכילה ייצוג טקסטואלי של אותו העובד: מקום עבודה ומשרה.

חלק ב (10 נקודות )

כתבו מחלקת ClassMain אשר תכיל את המטודות הבאות:

1. menu() אשר תציג למשתמש את התפריט הבא:

[1] –Student

[2] - Employee

[3] – StudentEmployee

הפונקציה תקבל מהמשתמש בחירה 1 2 או 3 ותחזיר את תוצאת הבחירה.

במידה והמשתמש הקליד מספר שהוא לא 1 2 או 3 התוכנית תחזיר הודעת חריגה exception .

1. Public static void main (String[] args)

ה main יקלוט מהמשתמש את כל נתוני האנשים לתוך מערך גנארי arraylist, במידה והמשתמש יכניס תחת תעודת זהות 1- התוכנית תפסיק את הקליטה. שימו לב כי יש להשתמש בפונקציה מסעיף 1 בכדי לדעת את סוג הסטודנט שיש להכניס למערך (זה משנה כי לכל סוג סטודנט יש בנאי שונה ).

1. כתבו פונקציה isEmployee (ArrayList<Person> arr) אשר מדפיסה את כמות העובדים (סטודנט לא בהכרח עובד ).

**הערות לשני החלקים:**

* יש להגדיר בנאים, gets/sets לכל משתני המחלקות
* ניתן להגדיר מחלקות או\ו ממשקים נוספים במידת הצורך.
* ניתן לקבוע עץ הורשה ( extends or implements ) לפי שיקול דעתך.
* עליך לדאוג שקוד של פונקציות ימומש גבוה ככל הניתן בהיררכית המחלקות ולא ישתכפל. כתוב קוד כך שהאוספת מחלקות נוספות לא תגרום לשינוי של קוד הקיים.

**שאלה 3 ( 30 נקודות ):**

כתוב מחלקה בשם Transaction המייצגת טרנזקציה על חשבון בנק.   
טרנזקציה על חשבון בנק היא בעצם פעולה בנקאית שיש לבצע בחשבון:

1. בנאי (Ctor) המקבל שלושה פרמטרים :
2. מספר חשבון בן 8 ספרות
3. ערך האם החשבון הוא עסקי או פרטי.  
   יש לממש את החשבון הפרטי או העיסקי באמצעות enum
4. יתרה נוכחית – סכום הכסף הקיים בעו"ש.
5. Deposit – פונקציה המפקידה כסף בחשבון. הפונקציה מקבלת פרמטר אחד בלבד והוא סכום הכסף שיש להפקיד בחשבון. הפונקציה שומרת את סכום הכסף כ – member/attribute של המחלקה ולא מבצעת בפועל את ההפקדה.
6. Pull - פונקציה המושכת כסף בחשבון. הפונקציה מקבלת פרמטר אחד בלבד והוא סכום הכסף שיש למשוך מהחשבון. הפונקציה שומרת את סכום הכסף כ – member/attribute של המחלקה ולא מבצעת בפועל את המשיכה.
7. Execute – פונקציה המבצעת בפועל את ההפקדה או המשיכה ומעדכנת את היתרה.
8. Balance – פונקציה המחזירה את היתרה בחשבון בכול שלב. גם לפני ביצוע הטרנזקציה בפועל וגם אחרי.
9. חשבון יכול להיות במינוס(יתרת חובה) או בפלוס(יתרת זכות).
10. אין צורך ליצור בנאי ריק.
11. אם הסכום להפקדה או משיכה הוא אפס או שלילי הפונקציה תוציא חריגה(Exception) ללא קריסה של התוכנית.

+ התוכנית צריכה לתמוך בעד 10000 טרנזקציות.

+ צור 2 Threadים העוברים על התור ומבצעים את הטרנזקציות:

1. Thread ראשון העובר על התור ומטפל רק בלקוחות פרטיים.
2. Thread שני העובר על התור ומטפל בלקוחות עסקיים ופרטיים.

**בהצלחה !**