**פתרון Y**

**שאלה 1 (50 נקודות)**

בשאלה זו ננהל נתונים של תחרות כלבים, ולכן נעבוד עם 3 מחלקותDog , ExercisesGrade ו- DogContest.

הקבצים נמצאים ב package question\_1.

המחלקהDog : שכרגע מכילהenum עם סוגי כלבים, קבוע לשם ברירת מחדל לכלב, קבוע מספר התרגילים המקסימאלי שכלב יכול להתחרות בהם, שדה לשם הכלב, מספר ת"ז של הכלב, מספר הניצחונות בעבר של הכלב, מספר ההפסדים של הכלב, ואובייקט מסוג ExerciseGrades . בנוסף, יש שיטה לשמירת האובייקט לקובץ וחתימה של שיטה getName() )שאינה ממומשת).

המחלקהExerciseGrades : מתארת ציונים, השדות שלה הם ציון בתרגיל 1 וציון בתרגיל 2.

המחלקה: DogContest מכילה כרגע, קבוע עם מספר הכלבים המקסימלי, מערך הכלבים, ומספר הכלבים בפועל.

בשאלה זו לא נתון main. עליכם ליצור אותו בעצמכם ולבדוק את תקינות התכנית (לא חובה, אבל רצוי).

1. צרו מחלקת **שירות** StringCheck לטיפול במחרוזות לפי ההגדרות בסעיף זה
   1. (4 נק׳) ממשו את השיטה boolean isStartWithCapital(String str) אשר בודקת האם המחרוזת מתחילה באות גדולה
   2. (3 נק׳) ממשו את השיטה findChar(String str,char findMe) אשר מחזריה את האינדקס הראשון בstr ששווה לתו findMe אם התו לא נמצא במחרוזת השיטה תחזיר קבוע שיסמן זאת.
   3. (3 נק׳) ממשו את השיטה boolean onlyLetters(String str) אשר תבדוק אם str מכילה רק תווים שהם אותיות באנגלית.
2. המחלקה ExerciseGrade
   1. (2 נק׳) ממשו בנאי המקבל Scanner פתוח לקובץ ומיצר את השדות . ממשו

ExerciseGrade(Scanner scan)  
מבנה הקובץ קלט:

100

98

כלומר, בתרגיל מספר 1 ניתן ציון 100 ובתרגיל 2 ניתן ציון 98

על הקלט הנל הרצת toString תניב  
 gradeOne=100, gradeTwo=98

2.2 (2 נק׳) ממשו במחלקה ExerciseGrade שיטה לשמירה בקובץ. אשר מקבלת אובייקט מוכן לכתיבה (קובץ פתוח) ותכתוב בשורה הראשונה את הציון הראשון ובשניה את השני .

1. הרחיבו את המחלקה Dog
   1. (1 נק') עדכנו במחלקה Dog את השיטה getName();
   2. (3 נק׳) ממשו את השיטה getOddsToWin() אשר מחזירה את היחס בין מספר הניצחונות בעבר לכול התחרויות כמספר בין 0 ל-1. לדוגמא, כלב שניצח 4 פעמים (המשתנה pastWin) והפסיד 4 פעמים (המשתנה pastLost ) נקבל תשובה 0.5 ( )
   3. (3 נק׳) ממשו בנאי למחלקה המקבל כפרמטרים את שם הכלב, מספר הניצחונות ,מספר ההפסדים בעבר, והסוג כלומר ממשו

public Dog(String name, int pastWin, int pastLost, dogType type)

* 1. (4 נק׳) לכל כלב יש id יחיד שמאפיין אותו, עדכנו את הבנאי שיתמוך בתכונה זאת (כלומר כל כלב ייווצר עם id ייחודי לו) אתם כמובן רשאים להוסיף שיטות ומשתנים למחלקה.
  2. (5 נק׳) שם של כלב מורכב משם המשפחה קו תחתון ואז שם הכלב . שם הכלב ושם המשפחה חייבים להתחיל באות גדולה . השם יכיל רק קו תחתון אחד ואותיות אנגליות בלבד. בדקו זאת. אם אחד התנאים לא מתקיים שמו של הכלב יהיה Sal\_D כקבוע במחלקה. **השתמשו במחלקת השירות שכתבתם בסעיף 1**
  3. (3 נק) כלב מחזיק שדה מסוג ExercisesGrade. צרו בנאי נוסף שמקבל גם ערכים של שני ציונים ומאתחל כלב עם השדה Grades מאותחל כלומר ממשו.

Dog(String name, int pastWin, int pastLost, dogType type, int grade1,int grade2)

* 1. (3 נק׳) ממשו toString יש שני מקרים 1 האובייקט ExerciseGrade לא אותחל ואז הפלט יהיה

name=Sal\_D , id=0 , pastWin=4 , pastLost=4 , oddsToWin=0.5 , type=Milinoar

ואם האובייקט ExerciseGrade קיים הפלט יהיה:

name=Jon\_Snow, id=1, pastWin=2, pastLost=6, type=Milinoar, grades: gradeOne=90, gradeTwo=98

הקפידו על reuse של קוד.

* 1. ( 4 נק) ממשו בנאי הקורא את המידע מתוך קובץ לפי הדוגמה הנ"ל ועובד עם Scanner מוכן כלומר ממשו public Dog(Scanner scan)

קלט:

Sal\_D

4

6

Milinoar

2

95

98

שימו לב, במחלקה Dog יש שיטהsave אשר בונה את הקלט לבנאי על הבנאי להתאים בדיוק לפלט של Save

1. המחלקה DogContest מכילה קבוע שהוא מספר הכלבים המקסימאלי ומספר כמה כלבים כבר נוספו לתחרות. בנוסף לשיטה toString וחתימה של שיטה getWinner שתממשו בהמשך   
   1. (1 נק ) כתבו בנאי ריק כלומר בנאי שלא מקבל פרמטרים.
   2. )3 נק) ממשו את השיטה boolean addDog(Dog dog) אשר מנסה להוסיף כלב לתחרות.
   3. (5 נק') כתבו בנאי המקבל שם קובץ וקורא את נתוניו מהקובץ. מבנה הקובץ יוצג בהמשך. העזרו בדוגמת הקובץ המצורף ובהסברים בהמשך כדי להבין את סדר קריאת הערכים.
   4. (4 נק ) מצאו מי הכלב עם סיכוי הניצחון הגבוה ביותר מבין כל הכלבים והחזירו אותו   
      ממשו Dog getFavoriteDogToWin()
   5. (5 נק ) מצאו מי הכלב שניצח בתחרות – ניצחון מוגדר לכלב עם **סכום הציונים בתרגילים גבוה ביותר**. כלומר, כלב שהתחרה בשני תרגילים וקיבל 60 ו 80 ינצח כלב שהתחרה בתרגיל 1 וקיבל 100 ו-30 בתרגיל השני. ניתן להוסיף שיטות למחלקות אחרות.  
      ממשו אתDog getWinner()

קלט לדוגמה

המספר הראשון (4) מיצג שיש 4 כלבים אחר כך כתובים 4 כלבים בעזרת שיטת הכתיבה של DOG

4

Sal\_D

4

6

Milinoar

0

100

100

Girl\_HasNoName

6

2

Boxer

0

28

90

Dany\_QueenOf

2

6

Boxer

1

42

42

Jhon\_Snow

5

6

Chowchow

2

2

5

נתון קובץ בשםdogTur.txt שבעזרתו תוכלו לבדוק את המחלקות שכתבתם. יש לוודא שהפלט של toString יוצא בדיוק כמו הפלט הבא:

The dog contest has the following dogs :

name=Sal\_D, id=2, pastWin=6, pastLost=0, type=Milinoar, grades: gradeOne=100, gradeTwo=100

name=Girl\_HasNoName, id=0, pastWin=6, pastLost=2, type=Boxer, grades: gradeOne=28, gradeTwo=90

name=Dany\_QueenOf, id=1, pastWin=2, pastLost=6, type=Boxer, grades: gradeOne=42, gradeTwo=42

name=Jhon\_Snow, id=3, pastWin=5, pastLost=6, type=Chowchow, grades: gradeOne=2, gradeTwo=5

The wining dog is Sal\_D

**פתרון**

**package** question\_1;

**import** java.io.File;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Scanner;

**public** **class** Dog {

**public** **enum** DogType {

***Boxer***, ***Milinoar***, ***Poodale***, ***Chowchow***

};

**public** **static** **final** **int** ***MAX\_EXERCISE***=5;

**private** **final** String DOG\_NO\_NAME="Sal\_D";

**private** **static** **final** **int** ***DOG\_HAS\_NO\_GRADES*** = -1;

**private** **static** **int** *idGen* = 0;

**private** String name;

**private** **int** pastWin;

**private** **int** pastLost;

**private** DogType type;

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

**private** ExercisesGrade grades;

**public** Dog(String name, **int** pastWin, **int** pastLost, DogType type) {

**if**(!checkName(name)) {

name=DOG\_NO\_NAME;

}

**this**.name = name;

**this**.pastWin = pastWin;

**this**.pastLost = pastLost;

**this**.type = type;

**this**.id = *idGen*++;

}

**public** Dog(String name, **int** pastWin, **int** pastLost, DogType type, **int** gradeOne, **int** gradeTwo) {

**this**(name,pastWin,pastLost,type);

grades=**new** ExercisesGrade(gradeOne, gradeTwo);

}

**public** Dog(Scanner scan) {

name = scan.next();

pastWin = scan.nextInt();

pastLost = scan.nextInt();

type = DogType.*valueOf*(scan.next());

id=scan.nextInt();

grades=**new** ExercisesGrade(scan);

}

**public** **static** **int** getMaxExercise() {

**return** ***MAX\_EXERCISE***;

}

**public** **int** getId() {

**return** id;

}

**public** String toString() {

String back="name=" + name + ", id=" + id+ ", pastWin=" + pastWin + ", pastLost=" + pastLost + ", type=" + type ;

**if**(grades!=**null**) {

back=back+", grades: " + grades;

}

**return** back;

}

**public** **double** getOddesToWin() {

**double** sum = pastLost + pastWin;

**return** pastWin / sum;

}

**public** String getName() {

**return** name;

}

**private** **boolean** checkName(String name) {

**int** underLine=StringCheck.*findChar*(name,'\_');

**if**(underLine==StringCheck.***CHAR\_NOT\_FOUND***) {

**return** **false**;

}

String dogName=name.substring(underLine+1);

String famName=name.substring(0,underLine);

**if**(!StringCheck.*isStartWithCapital*(dogName)||(!StringCheck.*isStartWithCapital*(famName))) {

**return** **false**;

}

**if**(!StringCheck.*onlyLetters*(dogName)||(!StringCheck.*onlyLetters*(famName))) {

**return** **false**;

}

**return** **true**;

}

**public** **void** save(PrintWriter writer) **throws** IOException {

writer.println(name);

writer.println(pastWin);

writer.println(pastLost);

writer.println(type.name());

writer.println(id);

writer.println(grades.getGradeOne());

writer.println(grades.getGradeTwo());

}

**public** **int** getTotalGrades() {

**if**(grades==**null**) {

**return** ***DOG\_HAS\_NO\_GRADES***;

}

**return** grades.getTotal();

}

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

Dog d = **new** Dog("d", 4, 6, DogType.***Milinoar***,100,100);

Dog d2 = **new** Dog("Jon\_Snow", 2, 6, DogType.***Milinoar***,90,98);

System.***out***.println(d.getOddesToWin());

System.***out***.println(d2);

System.***out***.println(d);

System.***out***.println(d);

PrintWriter writer = **new** PrintWriter("dog1");

d.save(writer);

writer.close();

// File f = new File("dog1");

// Scanner s = new Scanner(f);

// Dog d2 = new Dog(s);

// System.out.println(d2);

// Dog d1 = new Dog("d11", 14, 6, dogType.Milinoar);

// System.out.println(d1);

}

}

**package** question\_1;

**import** java.io.File;

**import** java.io.FileNotFoundException;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Scanner;

**import** question\_1.Dog.DogType;

**public** **class** DogContest {

**private** Dog[] dogs;

**public** **final** **static** **int** ***MAX\_NUMBER\_OF\_DOGS*** = 5;

**private** **int** numberOfDogs;

**public** DogContest() {

numberOfDogs = 0;

dogs = **new** Dog[***MAX\_NUMBER\_OF\_DOGS***];

}

**public** DogContest(String fileName) **throws** FileNotFoundException {

**this**();

File file = **new** File(fileName);

Scanner scan = **new** Scanner(file);

numberOfDogs = scan.nextInt();

**for** (**int** i = 0; i < numberOfDogs; i++) {

dogs[i] = **new** Dog(scan);

}

}

**private** **void** save(String fileName) **throws** IOException {

File file = **new** File(fileName);

PrintWriter writer = **new** PrintWriter(file);

writer.println(numberOfDogs);

**for** (**int** i = 0; i < numberOfDogs; i++) {

dogs[i].save(writer);

}

writer.close();

}

**public** **boolean** addDog(Dog dog) {

**if** (numberOfDogs >= dogs.length) {

**return** **false**;

}

dogs[numberOfDogs++] = dog;

**return** **true**;

}

**public** Dog getFavoriteDogToWin() {

**int** bestIndex = 0;

**for** (**int** i = 1; i < numberOfDogs; i++) {

**if** (dogs[i].getOddesToWin() > dogs[bestIndex].getOddesToWin()) {

bestIndex = i;

}

}

**return** dogs[bestIndex];

}

**public** Dog getWinner() {

**int** bestIndex = 0;

**for** (**int** i = 1; i < numberOfDogs; i++) {

**if** (dogs[i].getTotalGrades() > dogs[bestIndex].getTotalGrades()) {

bestIndex = i;

}

}

**return** dogs[bestIndex];

}

**public** String toString() {

StringBuffer str = **new** StringBuffer("The dog contest has the following dogs : ");

**for** (**int** i = 0; i < numberOfDogs; i++) {

str.append("\n");

str.append(dogs[i].toString());

}

str.append("\nThe wining dog is ");

str.append(getWinner().getName());

**return** str.toString();

}

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

// DogContest tournament=new DogContest("dogTur.txt");

DogContest tournament = **new** DogContest();

Dog d3=**new** Dog("Girl\_HasNoName",6,2,DogType.***Boxer***,28,90);

Dog d4=**new** Dog("Dany\_QueenOf",2,6,DogType.***Boxer***,42,42);

Dog d = **new** Dog("d",6,0,DogType.***Milinoar***,100,100);

tournament.addDog(d);

tournament.addDog(d3);

tournament.addDog(d4);

Dog d1 = **new** Dog("Jhon\_Snow", 5, 6, DogType.***Chowchow***, 2, 5);

tournament.addDog(d1);

tournament.save("dogTur.txt");

//System.out.println(tournament);

DogContest t1=**new** DogContest("dogTur.txt");

System.***out***.println(t1);

}

}

**package** question\_1;

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** java.util.Scanner;

**public** **class** ExercisesGrade {

**private** **int** gradeOne;

**private** **int** gradeTwo;

**public** ExercisesGrade(**int** gradeOne, **int** gradeTwo) {

**this**.gradeOne = gradeOne;

**this**.gradeTwo = gradeTwo;

}

**public** ExercisesGrade(Scanner scan) {

gradeOne=scan.nextInt();

gradeTwo=scan.nextInt();

}

**private** **void** save(PrintWriter writer) **throws** IOException {

writer.println(gradeOne);

writer.println(gradeTwo);

}

**public** String toString() {

**return** "gradeOne=" + gradeOne + ", gradeTwo=" + gradeTwo ;

}

**public** **int** getGradeOne() {

**return** gradeOne;

}

**public** **int** getGradeTwo() {

**return** gradeTwo;

}

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

ExercisesGrade ex =**new** ExercisesGrade(100,98);

System.***out***.println(ex);

}

**public** **int** getTotal() {

// **TODO** Auto-generated method stub

**return** gradeOne+gradeTwo;

}

}

**package** question\_1;

**public** **class** StringCheck {

**public** **static** **final** **int** ***CHAR\_NOT\_FOUND***=-1;

**public** **static** **boolean** onlyLetters(String str) {

**for** (**int** i = 0; i < str.length(); i++) {

**if**(!Character.*isLetter*(str.charAt(i))){

**return** **false**;

}

}

**return** **true**;

}

**public** **static** **int** findChar(String str,**char** findMe) {

**for** (**int** i = 0; i < str.length(); i++) {

**if**(str.charAt(i)==findMe) {

**return** i;

}

}

**return** ***CHAR\_NOT\_FOUND***;

}

**public** **static** **boolean** isStartWithCapital(String str) {

String cap=str.toUpperCase();

**char** first=cap.charAt(0);

**if**(first<'A'||first>'Z') {

//not a letter

**return** **false**;

}

**return**(first==str.charAt(0));

}

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

System.***out***.println(*onlyLetters*("rtrtFFr"));

}

}

**שאלה 2 (30 נקודות)**

הקוד נמצא בחבילה question\_2. שימוש בלולאות יגרור פסילת הפתרון.

מספר מאושר הוא מספר אשר אם מחברים את סכום ריבועי ספרותיו בתהליך חוזר מקבלים את המספר 1.

מספרים מאושרים לדוגמא: 13 ,19, 32 , 82 , 100

דוגמאות לחישוב עבור המספר 19:

12 + 92=82  
82 + 22=68  
62 + 82=100  
12 + 02 + 02=1

מספר מאושר מתקבל בהכרח תוך 8 איטרציות או פחות.

בשאלה זו נכתוב תכנית המציגה את כל המספרים המאושרים מ - 10 ועד שיש 3 מספרים רצופים שהם מאושרים, לטובת המימוש יש לכתוב ולהשתמש בפונקציות הבאות:

1. (10 נקודות) כתוב פונקציה רקורסיבית המקבלת מספר ומחזירה את סכום ריבועי הספרות המרכיבות אותו. לדוגמא, עבור 68 : 62 + 82=100

**public** **static** **int** sumDigitsPower(**int** num)

1. (10 נקודות) כתוב פונקציה רקורסיבית המקבלת מספר ומקסימום חזרות (איטרציות) ומחזירה האם המספר הוא מאושר או לא, השתמש בפונקציה מסעיף א. 68 הוא מספר מאושר.

**public** **static** **boolean** isHappyNumber(**int** num, **int** numOfRetries)

1. כתוב פונקציה רקורסיבית המקבלת מספר שממנו צריך להתחיל את החיפוש, ומספר המספרים העוקבים שנדרש למצוא. הפונקציה מדפיסה 3 מספרים מאושרים עוקבים.  
   היעזר בקבוע שמוגדר ב Starter, השתמש בפונקציות בסעיפים א ו ב. הפלט הנדרש הוא: "*1880,1881,1882*"

**public** **static** **boolean** print3ConsecutiveHappyNumbers(**int** fromNum, **int** numOfConsecutivesToFind)

**פתרון**

**package** question\_2;

**public** **class** Question2 {

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

// **TODO** Auto-generated method stub

System.***out***.println(*sumDigitsPower*(68));//100

System.***out***.println(*isHappyNumber*(68, 8));//true

*print3ConsecutiveHappyNumbers*(100, 3);//1880,1881,1882

}

**public** **static** **void** print3ConsecutiveHappyNumbers(**int** fromNum, **int** numOfConsecutivesToFind) {

**final** **int** NUM\_OF\_CONSECUTIVES = 3;

**if** (numOfConsecutivesToFind > 0) {

**if** (*isHappyNumber*(fromNum, 8)) {

numOfConsecutivesToFind--;

} **else** {

numOfConsecutivesToFind = NUM\_OF\_CONSECUTIVES;

}

fromNum++;

*print3ConsecutiveHappyNumbers*(fromNum, numOfConsecutivesToFind);

} **else** {

System.***out***.println((fromNum-3) + "," + (fromNum-2) + "," + (fromNum-1));

}

}

**public** **static** **boolean** isHappyNumber(**int** num, **int** numRetries) {

**int** tempResult = 0;

**if** (num > 1 && numRetries != 0) {

tempResult = *sumDigitsPower*(num);

numRetries--;

**return** *isHappyNumber*(tempResult, numRetries);

} **else** **if** (num == 1) {

**return** **true**;

} **else** {

**return** **false**;

}

}

**public** **static** **int** sumDigitsPower(**int** num) {

**int** result = 0;

**int** modulo;

**if** (num > 0) {

modulo = num % 10;

num = num / 10;

result = (**int**) Math.*pow*(modulo, 2);

**return** result + *sumDigitsPower*(num);

} **else** {

**return** result;

}

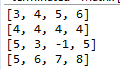
}

}

**שאלה 3 (20 נקודות)**

יש לכתוב תכנית המקבלת מטריצה של מספרים שלמיםmat בגדול n על n ומערך list בגודל n של מספרים שלמים **שונים זה מזה**. הפונקציה בודקת אם יש שורה במטריצה המכילה בדיוק את אותם מספרים כמו המערך. נסו לכתוב קוד יעיל! קוד בסיבוכיות מעל יקבל ניקוד חלקי . ניתן לפתור ביעילות . אין צורך לבדוק את תקינות הקלט. יש לענות ב קובץ Question3.java אשר נמצא בחבילה question\_3. יש שם רק main ריק, עליכם להבין מה חתימת הפונק׳ ולדאוג לקריאה.

דוגמה, עבור המטריצה:



* עבור המערך 3,5,4,6 נקבל תשובהfound in line 1
* עבור המערך 6,7,5,8נקבל תשובהfound in line 3
* עבור המערך 7,42,2,8 נקבל תשובהnot found

**פתרון:**

**package** question\_2;

**public** **class** Question2 {

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

// **TODO** Auto-generated method stub

System.***out***.println(*sumDigitsPower*(68));//100

System.***out***.println(*isHappyNumber*(68, 8));//true

*print3ConsecutiveHappyNumbers*(100, 3);//1880,1881,1882

}

**public** **static** **void** print3ConsecutiveHappyNumbers(**int** fromNum, **int** numOfConsecutivesToFind) {

**final** **int** NUM\_OF\_CONSECUTIVES = 3;

**if** (numOfConsecutivesToFind > 0) {

**if** (*isHappyNumber*(fromNum, 8)) {

numOfConsecutivesToFind--;

} **else** {

numOfConsecutivesToFind = NUM\_OF\_CONSECUTIVES;

}

fromNum++;

*print3ConsecutiveHappyNumbers*(fromNum, numOfConsecutivesToFind);

} **else** {

System.***out***.println((fromNum-3) + "," + (fromNum-2) + "," + (fromNum-1));

}

}

**public** **static** **boolean** isHappyNumber(**int** num, **int** numRetries) {

**int** tempResult = 0;

**if** (num > 1 && numRetries != 0) {

tempResult = *sumDigitsPower*(num);

numRetries--;

**return** *isHappyNumber*(tempResult, numRetries);

} **else** **if** (num == 1) {

**return** **true**;

} **else** {

**return** **false**;

}

}

**public** **static** **int** sumDigitsPower(**int** num) {

**int** result = 0;

**int** modulo;

**if** (num > 0) {

modulo = num % 10;

num = num / 10;

result = (**int**) Math.*pow*(modulo, 2);

**return** result + *sumDigitsPower*(num);

} **else** {

**return** result;

}

}

}