JAVA

cross platform - התוכנית ב JAVA יכולה לרוץ בהרבה מערכות הפעלה

open source - קוד פתוח

Performance - ביצועים, מהירות, תוקף שלו כל רגע נופלת

Multi threading - מתקי תהליכים יוצר לתהליכים שרצים במקביל

Garbage collection - מרוקן מהזכרון את כל האובייקטים הבינתיים

JVM - JAVA virtual machine - יוצר לתרגם שמה קובץ class.

wrapper - יש מחלקה לפרימיטיבים (int וכו')

מחלקה שעושה שהפרימיטיבים יתנהגו כמו אובייקט

boxing - כשהפך יוצר reference מעלרן הפרימיטיבי

unboxing - יוצר פרימיטיבי מה reference

object יורש - class extends

פוסק - פרוטוקול פשוט של JAVA

mutable - מושתק

אפשר לשנות מחוץ ומה לו מתבטל.

im-mutable - לא מושתק

אפשר לשנות.

execute - מפעיל את הבונה של האב super();

toString - באובייקט מחזיר את ה hashcode, אפשר לדרוס.

equals -

clone() - מעתיק מהמשתנה האמיתי

קיצור אפשר לשנות כל מיני דברים - alt + shift + s

setter / getter כמו

deprecated - המתודה

סוג של אזהרה שהיא הבונה של Java את יבא. אבל בbytecode יבוא לרוץ

control + o - open לכל method

Jre - הרחבה

bytecode - כמו קובץ exe ב C#

Window-Preferences - לבדוק לשנות נתים

alt + shift + l

MyProgram.java

```
;package lesson2

;import java.util.Date

} public class MyProgram

} public static void main(String[] args)

;Pupil pupil = new Pupil(1, "Miri", "Cohen", new Date())
;checkMutable(pupil)
;Pupil pupil2 = new Pupil(2, "Miri", "Levi")//
;(123)pupil.setId*/
;()pupil.getId
;pupil.setFirstName("Shoshana")
;()String otherName = pupil.getFirstName
/*;"otherName = "Shoshana1
;printPupil(pupil)//
;comparePupils(pupil, pupil2)//
;String a = new String("String")*/
;String b = new String("String")
}if(a == b)
;System.out.println("a and b are equals")
{
}else
;System.out.println("a and b are NOT equals")
/*{


{

}private static void checkMutable(Pupil pupil)
;()Date burnDate = pupil.getBurnDate
;(10)burnDate.setMonth
;System.out.println(pupil.getBurnDate().toLocaleString())
{

} private static void comparePupils(Pupil pupil, Pupil pupil2)

}if(pupil.equals(pupil2))
;System.out.println("pupils are same")
{
}else
;System.out.println("pupils are NOT same")
{
{

}public static void printPupil(Pupil pupil)
;System.out.println(pupil)
{

{
```

```
package lesson2;

import java.util.Date;

public class Pupil extends Object {

	private Integer id;
	private String firstName;
	private String lastName;
	private Date burnDate;
	//private Person person;

	public Pupil(int id, String firstName, String lastName, Date burnDate) {
		super();// execute the base constructor
		this.id = id;
		this.firstName = firstName;
		this.lastName = lastName;
		this.burnDate = burnDate;
	}

	@Override
	public String toString() {// base toString return the hashcode value, here we override to clear
value
		return "Pupil [id=" + id + ", firstName=" + firstName + ", lastName=" + lastName +
"]";
	}

	@Override
	public int hashCode() {
		final int prime = 31;
		int result = 1;
		result = prime * result + ((firstName == null) ? 0 : firstName.hashCode());
		result = prime * result + id;
		result = prime * result + ((lastName == null) ? 0 : lastName.hashCode());
		return result;
	}

	@Override
	public boolean equals(Object obj) {
		if (this == obj)
			return true;
		if (obj == null)
			return false;
		if (getClass() != obj.getClass())
			return false;

		Pupil other = (Pupil) obj;

		if (firstName == null) {
			if (other.firstName != null)
				return false;
		} else if (!firstName.equals(other.firstName))
			return false;

		return true;
	}
```

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

	public void setId(int id) {
		this.id = id;
	}

	public String getFirstName() {
		return firstName;
	}

	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}

	public String getLastName() {
		return lastName;
	}

	public void setLastName(String lastName) {
		this.lastName = lastName;
	}

	public Date getBurnDate() {
		return (Date) burnDate.clone();// implement im-mutable
	}

	public void setBurnDate(Date burnDate) {
		this.burnDate = burnDate;
	}

}
```

Person.java

```
package lesson2;

public class Person {

	int id;
	Integer idWrapper;
	boolean isValid;
	String name;

	public void setParameters(){
		id =1;
		idWrapper = new Integer(1);
		idWrapper = 1;	//boxing -- create a reference from the primitive value
		id = idWrapper.intValue();
		id = idWrapper;//un-boxing -- create a primitive from a reference

		name = idWrapper.toString();
	}

}
```

הכיני מחלקה של עובד עם הפרטים הבאים:

ת"ז

שם

משפחה

תאריך לידה

גובה

משקל

מתוכם:

primitive 2

wrapper 1

1 לממש immutable (להעזר במתודה clone)

לדרוס toString, equals

pojo - פרויקט של מחלקות פשוטות
↓
Project only java object

מחלקה object

כל המחלקות יורשות ממנה.

הפונקציות העיקריות - toString - מחזירה מחרוזת, - כתובת האובייקט (package),
hashcode - מחזיר משתנה מסוג int - יצרה מחוץ לטווח הנל
equals - מחזיר השוואת אובייקטים

יש חוק בJava שכאשר equals מחזיר true hashcode יחזיר לשניהם אובייקטים
אותו מספר מחוץ

final - אז לשנות מהמחלקה הבאה, פונקציות אז לדרוס אותן,
על שדה זה נקרא - אפשר לקבוע בשורה הראשונה, משתנה שיקבלו ערך אחד לשנות אותו
יציבה

המחלקה string

הוא final - immutable

משתמשים ב clone - אז לשנות, clone נעבור המחלקה אובייקט - הוא יוצר עותק בכתובת
mutiable - ניתן לשינוי
immutable - לא ניתן לשינוי

בכל עולם הכתבה יש בעיה בכל הנושא של reference, כאשר המחלקה מוקצה משתנה reference, ניתן לשנות
את הערך האורתוגרפי של ה reference בפונקציות שיקבלו אז אם הם עם getters ו setter private
המטודה get מחזירה מקבילה לאותו משתנה ולכן.

כשמשנה את הערך - בה מגדיר על אותו אובייקט והוא משתנה - בה mutiable - ניתן לשנות,
אין ניתן לעדכן ע"י התבנית של אובייקט אחר לעומת immutable - ניתן להשתמש בפונקציה clone
המחלקה string - כל פעם שקשור אליה הוא מקבלת מופע חדש ואלה הוא מעתיק את הstring
זה בעיה כי תופס הרבה מקום בזיכרון. לכן משתמשים ב stringBuilder - שמתאים בכה
כשמתמודדים מידי להתעסק עם ה string

קריאה וכתיבה

InputStream, OutputStream

שני מחלקות אבסטרקטיות שהן יורשות לכתיבה וקריאה

כדי ש throw exception - שגיאה, יש שגיאות קבועות כגון:

ה- FileOutputStream אם לא מוצאת את הקובץ היא אוקה שגיאה.

קורה שיש שגיאה מופע מיוחדת שאוכל את השגיאה לולא מקום שקרה לו, יש 2 אפשרות

אפשרות ראשונה הצהרה בלוגה - לזרוק גם את השגיאה

אפשרות שנייה try, catch, מנסה ב try אם יש שגיאה הולך לcatch,

אם רוצים שיקרה משהו בכל מקרה - נמקים ה- finaly

שבוצעה תמיד

exceptions

מעמיקים לסוגי השגיאות

ב JAVA יש מעמיקים לסוגי השגיאות, קיימת מחלקה exception שגיאה, להמלקה הזו קיימת מחלקות רבות שירשו אותה, כדי יש מקום בקוד שעלול לגרום שגיאה, לדוג בקריאה ולכתיבה אם הנתיב שנתן לו חוקי, אומנם נכתב לגרום שגיאה:

כדי לעשות כאילו בזמן הריצה (עלול לגרום שם שגיאה) נכתוב throw new exception

או שנגרום את אותה הירושה של המחלקה exception, נשים לתוך נתיבים שלנו בדומה של

ה exception לדוג- throw new FileNotFoundException("invalid file path")

יש לבדוק את המשמעות בתיעוד שעלולה לגרום שגיאה, כסף

שרוב הפקודות של המערכת נכתוב throws Exception אלא אם הירושה שלה.

יש לשים לב לבדוק מין throw new Exception לבין throws exception

throws exception - הכרזה על המערכת שהיא עלולה לגרום שגיאה

throw new exception - אותו דבר המשמעות בעצמה

כיצד מטפלים בשגיאות?

① נשים להכריז על המערכת שלנו גם שהיא יכולה שגיאות throws Exception-

② נשים לטפל בשגיאות ע"י try/catch - נעטוף את הקטע הבעייתי ב try, catch, אם

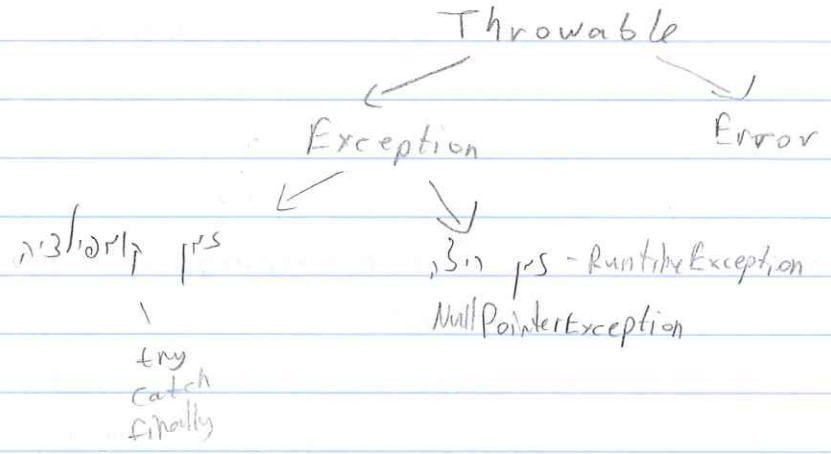
בתוך ה try יקרה בעיה בדרך כלל שגיאות בשפות תנסה לקלוט ה catch מהם

לא סוג ה- exception.

נוכל לכתוב בתוך try כמה catch-ים אבל לא סוג שגיאה

בתוך ה finally - כתוב תוכן שיקרה בכל מקרה גם אם נגרם שגיאה ותתבצע

נכנס ל block ה catch וגם אם ה try נגמר עד הסוף ללא שגיאות.



קיימים 3 סוגי שגיאות

1) שגיאות - צריך קומפילציה - checkedExceptions - שגיאות שהתוכנית יודעת עליהן ולכן
כדי קומפילציה יתובל לטפל בהם בקוד (או לצרוך זרוק ע"י try catch)

2) שגיאות - לא רלוונטי unchecked Exception - אלו שגיאות שאין עליהן התראה כדי קומפילציה
ושגיאות אלו מתרחשות בעקבות בעיות בקוד, שיגרום לו נתון ב-API לפי הרצוי
לדוגמה - בעת נסיון לחבור למקום לא קיים בזיכרון

3) Error - שגיאות כדי רלוונטי התוכנית שהתרחשו עקב בעיה בחומרה - לדוגמה אין
בזיכרון - OutOfMemoryError וכדומה, אלו errors לא נעשים ב-try catch כיוון שאין לה
שליטה בקוד

קריאה וכתיבה = I/O

נשתמש בספריה Java.io

Stream = זרם, רצף של נתונים

InputStream/OuputStream

InputStream = קריאה של נתונים ממקור חיצוני

OutputStream = כתיבה של נתונים למקור חיצוני

המחלקות הללו הן מחלקות אבסטרקטיות, קיימות להן מימושים שונים, לדוגמא:

FileInputStream/FileOutputStream

FileInputStream = קריאה של נתונים מקובץ

FileOutputStream = כתיבה של נתונים לקובץ

בבונה של המחלקות הללו יש להזין את הניתוב של הקובץ.

המחלקות הללו יודעות לקרוא ולכתוב רק ביטים.

דוגמא לכתיבת ביטים לקובץ ע"י FileOutputStream:

```
FileOutputStream out = new FileOuputStream("C:\\out.bin");
out.write(1);
out.close();
```

דוגמא לקריאת ביטים מקובץ ע"י FileInputStream:

```
FileInputStream in = new FileInputStream("C:\\out.bin");
while(in.read()!=-1){
in.read();
}
in.close();
```

BuffredInputStream/BufferedOutputStream

למחלקות InputStream ו-OuputStream ישנן מחלקות עוטפות שנקראות BufferedInputStream ו-BufferedOuputStream. המחלקות הללו גם כן קוראות וכותבות ביטים אולם להן יש יתרון כיוון שהן לא נגשות למקור הנתונים עבור כל ביט אלא הן שומרות אצלן קבוצה של ביטים ורק אז נגשות למקור הנתונים.

דוגמא לכתיבת ביטים לקובץ ע"י BuffredOutputStream:

```
FileOutputStream out = new FileOuputStream("C:\\out.bin");
```

```
BuffredOutputStream bos = new BuffredOuputStream(out);

bos.write(1);

bos.close();
```

דוגמא לקריאת ביטים מקובץ ע"י BuffreInputStream:

```
FileInputStream in = new FileInputStream("C:\\out.bin");

BuffredInputStream bis = new BuffredInputStream(in);

bis.read();

bis.close();
```

ObjectInputStream/ ObjectOutputStream

כדי לקרוא ולכתוב אוביקטים של ג'אווה:

1. יש לממש serializable – כדי לאפשר להפוך אוביקט לביטים יש לממש את ה- interface serializable במחלקות אותם נרצה לכתוב/לקרוא

לדוגמא:

```
public class Employee implements Seriablizable
```

2. נשתמש במחלקות הבאות:

ObjectInputStream

ObjectOutputStream

דוגמא לקריאת אוביקטים מקובץ ע"י ObjectInputStream:

```
ObjectInputStream ois = new ObjectInputStream(new BufferedInputStream(new
FileInputStream("C:\\file"));

Employee employee = ois.readObject();

ois.close();
```

דוגמא לכתיבת אוביקטים לקובץ ע"י ObjecOutputStream:

```
ObjectOutputStream oos = new ObjectOutputStream(new BufferedOutputStream(new
FileOutputStream("C:\\file");

oos.writeObject(employee)

oos.close();
```

תרגיל:

1. כתבי לקובץ את המספרים מ-1 עד 10

2. קראי את הקובץ והדפיסי אותו

3. הכיני מחלקה של עובד, מלאי את פרטיו בנתונים וכתבי אותו לקובץ

4. קראי את פרטי העובד מהקובץ והדפיסי אותם

```
1 package lesson4_1;
2 import java.io.FileNotFoundException;
3 import java.io.FileOutputStream;
4 import java.io.IOException;
5
6 public class FileOut {
7
8     public static void main(String[] args) {
9         FileOutputStream out;
10        try{
11            out = new FileOutputStream("D:\\sari.txt");
12            for(int i=0;i<10;i++)
13            {
14                out.write(i);
15            }
16            out.close();
17            }catch (FileNotFoundException e){
18                System.out.println("FileNotFoundException:" + e.getMessage());
19            } catch (IOException e) {
20                e.printStackTrace();
21            }finally{
22                System.out.println("Finish  Output");
23            }
24    }
25
26 }
27
```

```java
package lesson4_2;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class FileInput {

    public static void main(String[] args) {
        try{
        FileInputStream in;
            in = new FileInputStream("D:\\sari.txt");
            for(int i=0;i<10;i++){
            System.out.println(  in.read());
            }
            in.close();
    }catch (FileNotFoundException e) {
        System.out.println("FileNotFoundException:" + e.getMessage());
    } catch (IOException e) {
        e.printStackTrace();
    }
    finally{
        System.out.println("Finish Output");
    }

}
}
```

byte 型の
byte

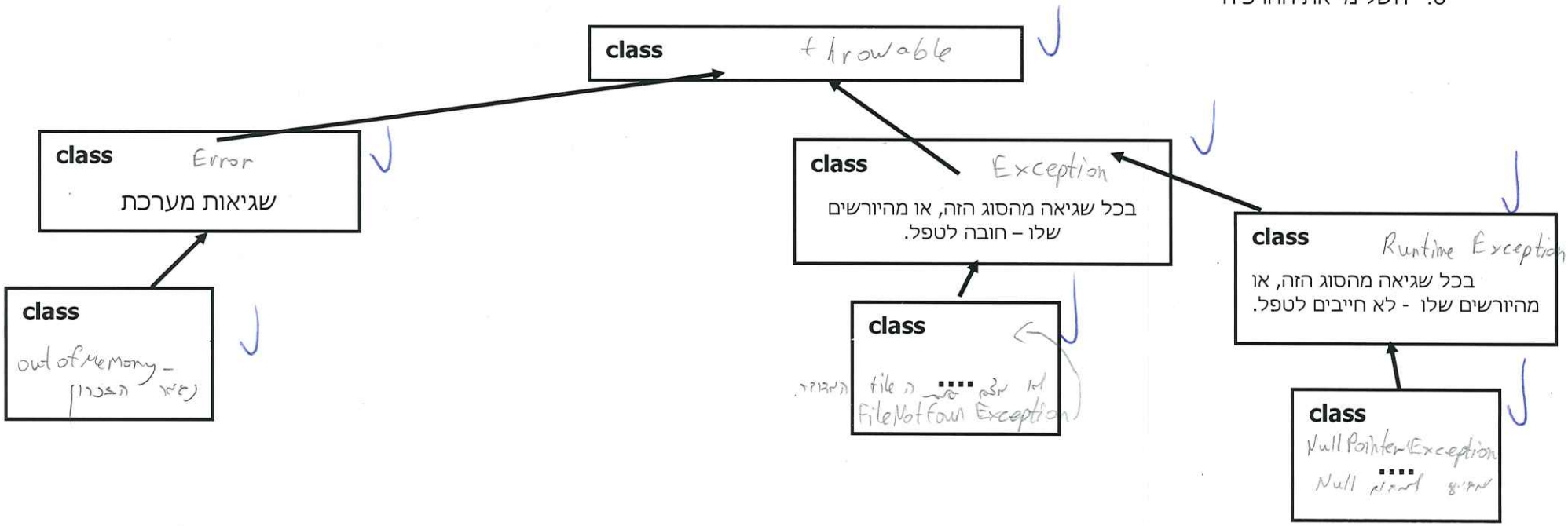
int x;
while ((x=infile.read())!=-1)
System.out.print(x);

1. מה ההבדל בין שגיאות קומפילציה לבין Exception? שגיאת קומפילציה - הבילת תקינות לפני הרצת הקוד. אם יש שגיאה אז לא יעבור לרצות. Exception אלו שגיאות שנופלות בזמן ריצה, אפשר לטפל בהן ב try ו catch
2. מה ההבדל בין checked exception ל- unchecked exception
3. האם הקטע הנ"ל תקין ומדוע? לא תקין, כי בקוד לא יכנס ל catch השני כי תמיד יזרוק את Exception.

```
        } catch (Exception exceptionObj) {
    //handle problem
    } catch (NullPointerException exceptionObj) {
            //handle problem
    }
```

7.5/9

4. איזה מידע ה exception מכיל?(3) מחלקה סוג היתה השגיאה, מחרוזת עם תיאור, מקום בקוד
5. מה ההבדל בין throws exception לבין throw exception? throws - מצביע. על המתודה שעלולה לזרוק, throw - פה הוא זורק את השגיאה.
6. מדוע לא נתפוס ב- catch שגיאה מסוג throwable? throwable יש מחלקות יורשות שאם אלו נמצאות ב try ו catch כמו ה-Error, לא נזרוק כי יורשות מ-Exception
7. כאשר יש return ב- catch האם ה finally מתבצע? - רשות כן, ה finally נקרא בכל מקרה.
8. השלימי את ההרכיה



שרי אלבין
ו/ב

Generics

Generic זוהי האפשרות ליצור מחלקה/מתודה כלליים שיתאימו לסוגים שונים של טיפוסים, לדוגמא: Date, String ועוד.

צורת הכתיבה:

- במחלקה גנרית: שם המחלקה ואח"כ <T>, לדוגמא:

```
public class GenericClass<T>
```

- במתודה גנרית: לפני סוג החזרה מציינים <T>, לדוגמא:

```
public <T> void genericMethod
```

מקובל לכתוב את הטיפוס הגנרי באות גדולה אחת, משתמשים בד"כ באות T=type, E=element

בעת הקצאת אוביקט מהמחלקה הגנרית נשלח ב-<> את הטיפוס הרצוי, לדוגמא:

```
GenericClass <String> obj
```

מכאן ואילך כל ההתייחסות במחלקה הגנרית ל- T תהיה לפי הסוג שנשלח.

יש אפשרות לכתוב גם <T extends ClassName/InterfaceName> – ואז ניתן לשלוח גם את כל היורשים/הממשים של המחלקה/הממשק שצוין.

בגרסאות קודמות של ג'אווה השתמשו במחלקה Object כדי ליצור גנריות.

יתרונות של שימוש ב- Generic לעומת שימוש ב- Object:

1. אכיפת הסוג – במחלקה הגנרית יאכף הסוג הנשלח למתודות עפ"י הסוג שצוין בהקצאת האוביקט, מה שאין כן ב- Object התומך בכל סוגי הטיפוסים.

2. חוסך המרות (- Casting) – המתודות הגנריות מחזירות מיד את הסוג הרצוי, מה שאין כן במתודות שמחזירות Object – ויש צורך אח"כ להמיר את הערך שחזר לסוג הרצוי.

הערה: כדי לתמוך בגרסאות קודמות של ג'אווה – בזמן הקומפילציה מתבצע Type Erasure – נמחק הסוג שנשלח בהקצאת האוביקט ובמקומו מוצב Object. מסיבה זו אי אפשר להפעיל מתודות המתייחסות לסוג המחלקה בזמן ריצה – כמו new T וכדו' – כיוון שבזמן ריצה הטיפוס הוא תמיד Object.

לדוגמא:

```
public class GenricClass<T> {

public void method (T item);

}

GenricClass<String> obj = new GenericClasss<>;
```

obj.method("hello"); - בזמן הקומפילציה יאכף המשתנה שנשלח ל-String בלבד כיוון שהוגדרה המחלקה String בהקצאה. אולם לאחר הקומפילציה המתודה תהיה כך:

```
public void method (Object item);
```

כיוון שמתבצע Type Erasure.

Wildcards:

כאשר נקצה אוביקט ממחלקה גנרית ונשלח סוג אב – לא נוכל להצביע עם ה-reference הזה על אוביקט מהמחלקה הגנרית ששלחו לה סוג בן, כיוון שהמטרה של Generic היא לאכוף שימוש של מחלקה אחת בלבד.

הפתרון במקרה כזה הוא להשתמש ב-Wildcards:

צורת ההקצאה של האוביקט מטיפוס האב תהיה:

GenricClass<? extends BaseClass> obj – הקצאה זו תואמת למחלקה BaseClass וכן לכל המחלקות היורשות אותה

או- GenericClass<? super ChildClass> obj – הקצאה זו תואמת למחלקה ChildClass וכן לכל המחלקות שמעליה בהיררכיית ההורשה.

יש אפשרות גם: GenericClass<?> obj – הקצאה זו תואמת לכל סוגי המחלקות.

הערה:

ב- Wildcards ניתן רק לאחזר את הנתונים היורשים – לעבור על הנתונים, אך לא לאתחל את הנתונים הנ"ל.

הרצאה

Generic - כללי - מקבלת מישהו "ערך" ואנשים מקבלים - מחלקים שונים מאוד.

מה לגבי היה טוב ם object שהיו בגרסה להשתמש ם Generic?

- עושה בדיקות קומפילציה שגורם המתכנת לא מסוגל שה Generic יהיה, ולשלם משהו אחר - תמיד יהיה. שומר על אמינות של סוג

בזמן הקומפילציה המתכנן בודק אם הנתונים של הד והנכון אם לא הד לסוג שבנתלה.

typeerase - כאשר המחשב מקמפל אם המחלקה לקוד byteCode, הוא הופך את ה-T של ה Generic ל object,

כמו is - instanceof - אפשר לעשות על Generic

wildcards - מישהו אחר שיוכל להכניס על כל הסוגים שהגדרנו מסוג T
לא מעניין אותנו, רק לדעת.

alt + shift + / - יכול משהו
alt + shift + r - שלנו לא לא שמלכים בדיוק

public class FindMax <T extends Comparable> - מקבל רק לא שווים

בס"ד

שם: שני סולדמן

קבוצה: 1/2

מבחן בג'אווה

סמני בכל אחת מהשאלות הבאות את האפשרות הנכונה / האפשרויות הנכונות.

יש לענות על 20 שאלות מתוך 22. צייני אילו שאלות הן שאלות רשות.

שימי לב- תתכן יותר מאפשרות נכונה אחת לשאלה!

1. איזו תכונה של ג'אווה מאפשרת לתכנית הכתובה בג'אווה לרוץ במערכות הפעלה שונות?

- Open Source
- Cross Platform
- Multi Threading
- Performance

2. אילו קבצים מריץ ה-JVM?

- קבצי bytecode
- קבצי java. src
- קבצי class. bin

3. מי מהטיפוסים הבאים הינו Wrapper?

- String
- float
- Pupil
- Integer
- Long
- Date

4. Integer x = 5; - האם תקין?

- כן, כי מופעל unboxing
- לא, כי Integer הוא Reference
- כן, כי מופעל boxing

5. מתי לא נשתמש ב- Primitive?

- ב- Collections
- ב- Generics
- בקריאה וכתיבה לקבצים
- כדי לאפשר Null

6. מהו הכלל בג'אווה לגבי המתודות hashcode, equals:

- אם hashcode מחזיר את אותו ערך לשני אוביקטים – גם equals יחזיר true בהשוואת שני האוביקטים
- אם equals מחזיר true בהשוואת שני אוביקטים – גם hashcode יחזיר את אותו ערך לשני האוביקטים

7. המתודה equals:

- משווה ערכים
- משווה כתובות
- משווה כתובות אלא אם כן מממשים אותה אחרת

8. משתנה מסוג final ניתן לאתחל:

- בשורת ההקצאה
- בבונה
- בבלוק הסטטי (-למשתנה סטטי) - כלומר שקראנו פעם אחת לכל האובייקטים כי זה לא אלו הסטטיים
- פעם אחת בלבד

9. המחלקה String היא:

- Mutable - ניתן לשינוי
- Immutable
- Final - שלא יוכלו לשנות אותה.

10. מהו הפלט עבור קטע הקוד הבא:

```
String s1 = new String("abc");
String s2 = new String("abc");
System.out.println(s1 == s2);
```

- true
- false

11. מהו הפלט עבור קטע הקוד הבא:

```
String s1 = new String("abc");
String s2 = new String("ABC");
System.out.println(s1 == s2);
```

- true
- false

12. לפנייך מימושים שונים למתודה getter של המשתנה Date date. מה מתוכם הוא Immutable?

- **return** date;
- **return** date.clone();
- **return new** Date(date.getTime());

13. מה ההשלכות של Type Erasure ב- Generics?

- יש להמיר את הטיפוס שחוזר לסוג הרצוי

- אין אפשרות לבצע new T
- יש להריץ תכנית הכוללת שימוש ב- Generics ב- JVM מגרסה 5 ומעלה בלבד

14. אלו משורות הקוד הבאות תקין – ללא שגיאות:

```
List<? extends Person> personList = new ArrayList<>();
```

wildcard - רק הוסיף - רק קשור לפרמטר

- personList.add(new Pupil());
- personList.add(new Person());
- Person person= personList.get(0);

15. מהי המשמעות של unchecked exception?

- שגיאות שאין עליהן התראה בזמן הקומפילציה
- שגיאות שאין עליהן התראה בזמן ריצה
- שגיאות שאין אפשרות לתקן אותן

16. כל שגיאה יורשת מ:

- Exception
- Throwable
- RuntimeException

17. מי מהטיפוסים הבאים הינו מחלקה?

- List
- ArrayList
- Collection
- LinkedList

18. באם לא נממש את המתודה equals במחלקה אותה נרצה לשמור ע"י HashSet:

- לא יהיו ערכים כפולים
- עלולים להיות ערכים כפולים
- יפגום בביצועים - יגיג יותר לאוט.

19. איזה Interface יש לממש בעת שימוש ב- TreeSet?

- Comparable
- Iterable
- Serializable

20. מה המשמעות של Serializable?

- ממשק ריק ללא משמעות
- מאפשר להפוך אוביקט של ג'אווה לביטים
- מאפשר לכתוב ביטים לקובץ

21. מתי נשתמש במחלקה DataOutputStream?

- כאשר נרצה לכתוב לקובץ את המספר 100
- כאשר נרצה לכתוב לקובץ את המספר 1000
- כאשר נרצה לכתוב לקובץ את האוביקט Pupil

22. כאשר יש שגיאת קומפילציה על השורה: **import** java.util, מה עלינו לעשות?

- יש לכוון את ה- eclipse ל- jdk המקומי כי ה- eclipse לא מוצא את ה- package: java.util
- יש לנסות להחליף את שם ה- package לשם תקין אחר, כי אין ספריה בשם java.util
- יש לפתוח workspace חדש

בהצלחה רבה בס"ד!

Collections

List - רשימה מסודרת

Set - אוסף ללא כפולים

Map - שומר put ולא add.

anonymous class

מחלקה אנונימית, משמשים כדי לעבוד, לא רואים את המשתנה המחלקה.

Predicate - מחזיר בוליאן
False / True

הלוך - collection לאיטרציה גם שאל את stream.
אפשר לבצע לעבור.

Java 8 - דרך מתכוננים לעבודה.

Tiles - שיבוט

Beans - רכיבים

JME
מכונה

JEE - multi threading

JSE - Java standard E

pure java

transaction - אי מעולה נבקיים אני רוצה
שלוש המעל

יש בדיוק שמעל הכלים - אז לעשות הכל
אינלמה

Jar - מחלקות של java שהן בארכיונים packages לא אפשר יפה מסמה
java archive

tomcat - לוקחים את הקוד שלי ב JEE ומריצה אותו (מינוע לכל דבר)
Server Application

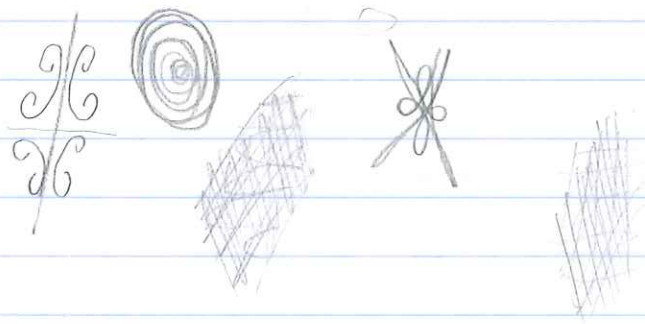
web.xml ב

ל הגדרות ב web-app

servlet - מחלקה בשרת שיודע לקבל קישור מ-http ומתבצע לעשות שלוש
מסתכלות על מידע ב request, ומחזיר response

request -

response - מידע שיוצא לנו



למחוק את המחלקה - alt + shift + x

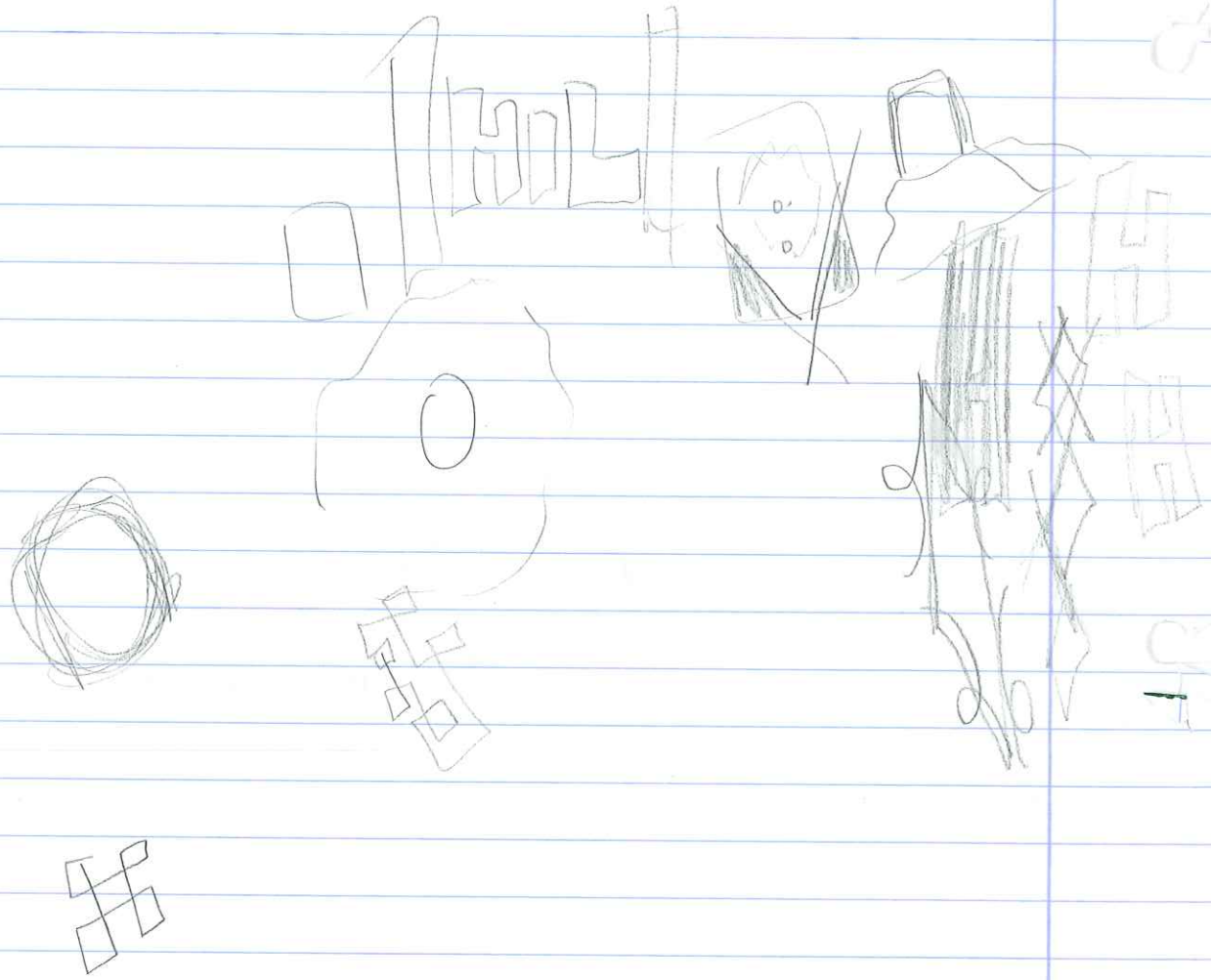
webservice - מנגנון שנותן ל client לבצע מדבר

filter - תרגום / controller -ו

ContainerResponseFilter - b דקשה שהיא / -

proxy - המשלח הנכון

ל מה לפני שיגיע
Response יכול לחזור



רק ב post - request payload

get - להביא נתונים יחידקטים תצוגה

post - ליצור

put - לעדכן

delete - למחיקה

Postman

מאפשר לבדוק קריאות אל web.API

שאינן דרך ה JAVA גם שנמצא אל

השלב הראשון להעלות את ה web.API ואחר כך להגיע אליו

domain - כתובת הסיום

הסבר	status
אין שום שרת לו (not found)	404
שגיאה בשרת הרשת נפל ולכן לא קיים)	500
ok, true	204
ok, true	200
	101
	304
method not allowed - אין תגובה לפי תוכנית. דרשו לתגובה דרך post שמגובה דרך get	405
	415

200-209

Date

- calendar - מחלקה מומלצת - לתפעל שנה, בקשה וכו' לכן יש מחלקה - Date מחלקת

```
calendar c = Calendar.getInstance
c.set(שנה, חודש, יום)
```

Local Date - מחלקה יותר נוחה ב- Java-8