**סיכום כללי – oop**

**הגדרות מונחים:**

**מחלקה** – **תבנית** ליצירת מופעים, המגדירה **טיפוס** שמכיל משתנים ופעולות.

ממחלקה אפשר ליצור מופעים על ידי המילה new והמופעים הנ"ל יווצרו בheap (יהיה להם מצביע)

**אובייקט** – מופע בודד של מחלקה (נוצר על ידי הבנאי)

**Class members** – פונקציות (פעולות) (בנאי הוא גם פונקציה) שמוגדרות בתוך המחלקה, ומשתנים (מאפיינים) שמוגדרים בתוך המחלקה

**הרשאת גישה** – הרשאות גישה יכולות להיות public / private ועוד, הגדרת המקומות מהם נוכל לגשת לclass member

**private** – הרשאת גישה , המגדירה את ההרשאה לגשת אל ה class member רק מתוך המחלקה הספציפית (לא דרך האובייקט, ולא דרך הנגזרת) – אם ניגש מחוץ למחלקה נקבל שגיאת קומפילציה

**public** - הרשאת גישה , המגדירה את ההרשאה לגשת אל ה class member מכל מקום (גם דרך המחלקה עצמה, גם דרך האובייקט, וגם דרך הנגזרת)

**static** – הגדרה ברמת המחלקה ולא ברמת האובייקט (המופע) הבודד, מאפשר לגשת ישירות לclass member הסטטי, ללא צורך ביצירת מופע

* מתוך המחלקה ניגש לclass member הסטטי דרך class name ולא דרך this
* ממחלקה יורשת ניגש לclass member הסטטי של הבסיס דרך class name ולא דרך this או super
* מחוץ המחלקה ניגש לclass member הסטטי דרך class name ולא דרך האובייקט

**Ctor –** בנאי – פונקציה דיפולטיבית הנמצאת בצורה אוטומטית בכל מחלקה שניצור.

על ידי האופרטור new נוכל לגשת לבנאי וליצור בheap אובייקט חדש מטיפוס המחלקה.

הבנאי יוצר בפועל מופע בזיכרון ומחזיר את הכתובת של המופע שיצר.

אם נגדיר בנאי שמקבל ערכים נהיה חייבים לשלוח את הפרמטרים בקריאה של הnew

**new –** אופרטור המשמש לקריאה לבנאי של מחלקה, לשם יצירת אובייקט מהמחלקה

**Composition** – הכלה של אובייקט ממחלקה A בתוך מחלקה B

**Inheritance** הורשה של מחלקה A למחלקה B (מחלקה B תמיד תכיל בתוכה את כל הclass members של מחלקת הבסיס)

* מחלקה A - מחלקת בסיס
* מחלקה - B מחלקת נגזרת
* יצירת אובייקט ממחלקה - B תקרא לבנאי של B ומשם בצורה מרומזת / מפורשת (ע"י super()) לבנאי של המחלקה A
* מהמחלקה הנגזרת אי אפשר לגשת לclass member המוגדרים כprivate בבסיס

**Multi level inheritance** - הורשה של מחלקה B שיורשת ממחלקה A למחלקה C (מחלקה C תמיד תכיל בתוכה את כל הclass members של A + B)

**Multi inheritance**—קבלת ירושה במחלקה C ממחלקה A וממחלקה B – דבר זה לא אפשרי בjava (וברוב שפות התכנות האחרות)

**Override** – דריסה של פונקציה ממחלקת הבסיס במחלקה הנגזרת. לשם כך יש לענות על הדרישות הבאות:

* הפונקציה בבסיס חייבת להיות עם אותו שם כמו הפונקציה בנגזרת
* הפונקציה בבסיס חייבת לקבל את אותו מספר פרמטרים שבפונקציה הדורסת (והפרמטרים חייבים להיות בעלי טיפוס זהה לפונקציה הדורסת)
* הפונקציה בבסיס חייבת להחזיר את אותו טיפוס שמוחזר מהפונקציה הדורסת

- **Abstract** מחלקה מופשטת (מחלקה בסיסית וראשונית ממנה נירש למחלקות ממשיות יותר), ממחלקה אבסרקטית אי אפשר ליצור מופעים, אבל אפשר להגדיר משתנה שיהיה מטיפוס המחלקה האבסרקטית ויוכל להצביע לכל אובייקט נגזרת (פולימורפיזם).

**Interface** – ממשק – מגדיר מעין "חוזה" שיכול להיות ממומש על ידי מחלקות

**Instanceof**- אופרטור המשמש לבדיקה:

* האם אובייקט מסוים הוא מופע של מחלקה מסוימת, או נגזרת של אותה מחלקה.

האופרטור יקבל בצד שמאל את שם האובייקט, ובצד ימין את שם המחלקה שנרצה לבדוק.

* האם אובייקט מסוים הוא מופע של מחלקה שמממשת ממשק מסוים

האופרטור יקבל בצד שמאל את שם האובייקט, ובצד ימין את שם הממשק שנרצה לבדוק.

**down casting** – המרה של אובייקט למחלקה אחרת (בד"כ ממחלקה בסיסית יותר למחלקה נגזרת)

**toString** – פונקציה שמוגדרת למחלקה object

**סיכום הרשאות והורשות:**

|  |  |  |  |
| --- | --- | --- | --- |
|  | מחלקה רגילה | מחלקה אבסטרקטית | ממשק |
| **הגדרת משתנה private** | כן | כן | לא |
| **הגדרת משתנה public** | כן | כן | Creates a final field (must be initialized) |
| **הגדרת משתנה ללא הרשאת גישה** | כן | כן | Creates a final field (must be initialized) |
| **הגדרת בנאי** | כן | כן | לא |
| **הגדרת פונקציה public כולל מימוש** | כן | כן | לא |
| **הגדרת פונקציה private כולל מימוש** | כן | כן | לא |
| **הגדרת פונקציהabstract private** | לא | לא | לא |
| **הגדרת פונקציהabstract public** | לא | כן | כן |
| **הגדרת פונקציה ללא מימוש וללא המילה abstract** | לא | לא | כן |
| **ירושה של מחלקה רגילה (extends)** | כן   * נוכל לגשת למשתנים לא סטטיים public של הבסיס על ידי המילה this * נוכל לגשת למשתנים סטטיים public של הבסיס על ידי שם מחלקת הבסיס * נוכל לגשת לפונקציות סטטיות public של הבסיס על ידי שם מחלקת הבסיס * נוכל לגשת לפונקציות לא סטטיות public של הבסיס על ידי המילה this או super – אם דרסנו את פונקציית הבסיס נוכל לפנות רק על ידי super | כן | לא |
| **ירושה של מחלקה אבסטרקטית (extends)** | כן   * חובה לממש את כל הפונקציות האבסטרקטיות * נוכל לגשת למשתנים לא סטטיים public של הבסיס על ידי המילה this * נוכל לגשת למשתנים סטטיים public של הבסיס על ידי שם מחלקת הבסיס * נוכל לגשת לפונקציות סטטיות public של הבסיס על שם מחלקת הבסיס * נוכל לגשת לפונקציות לא סטטיות public של הבסיס על ידי המילה this או super – אם דרסנו את פונקציית הבסיס נוכל לפנות רק על ידי super | כן   * לא חובה לממש את כל הפונקציות האבסטרקטיות – מה שלא מומש נהיה חייבים לממש בנגזרת * נוכל לגשת למשתנים לא סטטיים public של הבסיס על ידי המילה this * נוכל לגשת למשתנים סטטיים public של הבסיס על ידי שם מחלקת הבסיס * נוכל לגשת לפונקציות סטטיות public של הבסיס על שם מחלקת הבסיס * נוכל לגשת לפונקציות לא סטטיות public של הבסיס על ידי המילה this או super – אם דרסנו את פונקציית הבסיס נוכל לפנות רק על ידי super | לא |
| **מימוש של ממשק (implements)** | כן - חובה לממש את כל הפונקציות מהממשק  אפשר לממש כמה ממשקים במחלקה אחת | כן – אין חובה לממש את הפונקציות מהממשק (ואז חובה לממש אותם בנגזרת)  אפשר לממש כמה ממשקים במחלקה אחת | לא |
| **ירושה של ממשק (extends)** | לא | לא | כן – אפשר לרשת כמה ממשקים - וכאשר נממש במחלקה את הממשק הזה, נהיה חייבים לממש גם את כל מה שבממשק הבסיס |
| **הגדרת משתנה (מצביע)** | כן – יכול להצביע על אובייקטים מהסוג שלו או על אובייקטי נגזרת | כן – יכול להצביע רק על אובייקטי נגזרת | כן – יכול להצביע רק על אובייקטי מחלקה שמממשת את הממשק |
| **יצירת אובייקט (על ידי new)** | כן | לא | לא |
| **גישה על ידי מצביע בסיס למשתני נגזרת (שלא קיימים בבסיס)** | לא – חייב המרה מקומית לטיפוס הנגזרת | לא – חייב המרה מקומית לטיפוס הנגזרת | לא – חייב המרה מקומית לטיפוס הנגזרת |