**תרגול 10 OOP**

**Generics:**

* GENERICS מאפשר ליצור מחלקה, ממשק או מתודה שניתן להשתמש בה עם כל מיני סוגים של מידע (אובייקט) - נשים ב<> את סוג הטיפוס כשנשתמש על טיפוס ספציפי
* דרך לוודא ביטחון - טיפוס (type-safty)
* בעיה שיכולה לעלות בשימוש בGENERICS -
* אין קשר בין הטיפוסים השונים הנמצאים ב<>
* Class<string> לא יורשת מclass<object>
* עדיין ניתן להכניס מחרוזת למחלקה של OBJECT

**Wildcard:**

* דרך לשלוט יותר טוב על GENERICS
* WILDCARDS - כאשר נרצה להשתמש בטיפוס גנרי אבל אנחנו לא יודעים את הטיפוס הספציפי של הפרמטר ניתן להחליף אותו ב-?
* Placeholder ?
* אם נעשה את זה לא ניתן להניח שום דבר על הטיפוס של האובייקטים שנכניס
* עבור רשימה למשל - List<?>
* שימושי אם נרצה לעשות דברים ברמת הOBJECT
* למה להשתמש בזה?
  + למשל אם אנחנו רוצים להגדיר פונקציה שמדפיסה דברים אבל לא אכפת לנו איזה איברים יש ברשימה אז נוכל להשתמש ב?
  + לכל אובייקט יש את המחלקה toString
  + השוואת גודל של רשימות למשל - לא אכפת לנו מה יש ברשימות רק אם הגודל שווה
  + אם יש לנו פונקציה עם SWITCH שבכל CASE מחזירה רשימה עם טיפוסים שונים, אז נוכל להשתמש ב?
* כלומר השימושים הם עבור פעולות פשוטות מאוד שלא תלויות בטיפוס האיבר
* תכונות:
  + אנחנו לא יודעים מה הטיפוס ולכן לא נוכל להוסיף שום איבר חוץ מNULL
  + אם אנו עוברים על איבריה לא נוכל להניח עליהם כלום רק שהם OBJECT
  + רק רפרנסים יכולים להכיל wildcards - כלומר לא נוכח לייצר מחלקה או אובייקט קונקרטי שאנו לא יודעים מה הטיפוס שלו
  + מאפשרים לעשות up-casting פעמיים - גם על הטיפוס הגנרי וגם על המחלקה שהיא עצמה גנרית
  + List<?> l = new LinkedList<String>() - legal

**Bounding:**

* Upper bound - רק איברים שיורשים ממחלקה מסוימת - List<? Extends ClassA> ואז נוכל להתייחס לאיברים כרפרנסים של ClassA
* Lower bound - רק איברים שמחלקה מסוימת יורשת ממנו, שהוא SUPER של האיברים - List<? Super ClassB> - כל מי שהוא הורה של ClassB
* אי אפשר את שני החסמים
* יכול להיות על כל סוג של טיפוס - מחלקה, ממשק (גם במקרה זה נשתמש במילה extends)
* אם מדובר במשהו גנרי עם טיפוס מוגדר T נוכל להגיד extends T או super T
* שימוש בחסמים מאפשר לבצע פעולות ספציפיות שניתן לבצע על החסמים
* תכונות של חסם עליון:
  + ניתן לאתחל אותם עם כל דבר שיורש מהמחלקה המוגדרת כחסם
  + ניתן להוסיף רק Null
  + ניתן להתייחס לאיברים שם כרפרנסים של החסם ולכן נוכל להוציא אובייקט מהם כרפרנס של החסם
* תכונות חסם תחתון:
  + מאפשר להכניס דברים מהטיפוס של החסם (לא רק NULL)
  + כי יש לנו ברשימה איברים שהם בהיררכיה מעל החסם ולכן נוכל להכניס וההתייחסות אליהם ברשימה תהיה לפי הטיפוס שבהיררכיה מעל
  + לא מאפשר לקרוא את הדברים כי אנו לא יודעים מאיזה טיפוס הם
* איך זה מתבצע בJAVA? בדרך של **Erasure**
  + מוחק את כל הטיפוסים הגנריים בשלב הקומפילציה
  + הקומפיילר מוודא שלא עשינו משהו לא חוקי והופך למשל את List<String> ל-List
  + כי הוא כבר בדק את תקינות הטיפוסים
  + לאחר מכן כאשר יש מקומות בהם ניגשים לאיברים בתוך הרשימה הוא יודע שהוא יכול לעשות המרה לטיפוס המתאים כי הוא כבר בדק את תקינות הטיפוסים
* למה להשתמש בGENERICS?
  + תקינות טיפוסים - תהליך הerasure פועל רק אחרי שהקומפיילר מוודא את תקינות הטיפוסים, כלומר כאשר הקוד כבר type-safe
  + קריאות של הקוד - אנחנו לא רואים את השלב אחרי שהקומפיילר מוריד את הספציפיות
  + התהליך לא משפיע על היתרונות של שימוש בGENERICS, ומאפשר תמיכה בקוד עם ירושה.