**Chapter 2**

Creating and Destroying Objects

**Item 1**: Static factory მეთოდების გამოყენება კონსტრუქტორების ნაცვლად.

***public static Boolean valueOf(boolean b) {***

***return b ? Boolean.TRUE : Boolean.FALSE;***

***}***

**Advantages:**

1. ერთი უპირატესობა რაც ამ ტექნიკას აქვს არის ის რომ კონსტრუქტორებისგან განსხვავებით იყენებ მარტივად გასაგებ სახელებს.
2. კონსტრუქტორებისგან განსხვავებით არაა აუცილებელი რომ ამ მეთოდმა ყოველ ჯერზე ახალი ობიექტი შექმნას.
3. კონსტუქტორებისგან განსხვავებით შეუძლია დააბრუნოს ნებისმიერი სუბტიპი იმ კლასისა რომელსაც აბრუნებს.
4. დაბრუნებული ობიექტის კლასი შეიძლება განსხვავდებოდეს მეთოდიდან მეთოდზე იმ პარამეტრებზე დაყრდნობით რომლებსაც გადავცემთ.
5. დაბრუნებული ობიექტის კლასი აუცილებელი არაა რომ არსებობდეს მაშინ როცა ეს მეთოდი იწერება.

**Disadvantages:**

1. მთავარი ლიმიტაცია არის ის რომ ფაბლიქ/პროთექთიდ კონსტრუქტორის გარეშე, საბკლასის შექმნა არ გამოვა.
2. მეორე პრობლემა არის ის რომ ამ მეთოდების მოძებნა პროგრამისტებისთვის უფრო რთულია.

**Here are some common names for static factory methods:**

1. **From –** A type conversion method that takes a single parameter and returns a corresponding instance of this type.
2. **Of –** An aggregation method that takes multiple parameters and returns an instance of this type.
3. **valueOf –** A more verbose alternative to from and of.
4. **Instance or getInstance –** returns an instance that is described by its parameters but cannot be said to have the same value
5. **Create or newInstance –** Like instance or getInstance, except that the method guarantees that each call returns a new instance.
6. **getType –** Like getInstance, but used if the factory method is in different class. Type is the type of object returned by the factory method. For example ***FileStore fs = Files.getFilestore(path);***
7. **newType –** Like newInstance, but is used if the factory method is in different class. For example: ***BuferredReader bf = Files.newBuferredReader(path);***
8. **type -**  A concise Alternative to getType and newType. For example, ***List<Object> ls = Collections.list(params);***

**Item 2:** გამოიყენე Builder პათერნი როცა საქმე გაქვს ბევრ პარამეტრთან.

ბილდერის ალტერნატივები, მაშინ როცა ბევრი პარამეტრი გაქვს არის ტელესკოპის ფათერნი, როცა ჯერ ქმნი აუცილებელი პარამეტრების კონსტრუქტორს და შემდეგ ოპციური პარამეტრების კონსტრუქტორებს ცალ-ცალკე.

მეორე ალტერნატივა არის JavaBean ფათერნი, როცა გაქვს მხოლოდ ცარიელი კონსტრუქტორი და შემდეგ სეთერებით ცალ-ცალკე უცვლი. მაგრამ ამის პრობლემა არის ის, რომ JavaBean შეიძლება იყოს ინკონსისტენტურ მდგომარეობაში ამ კონსტრუქტორის გამოძახებისას.