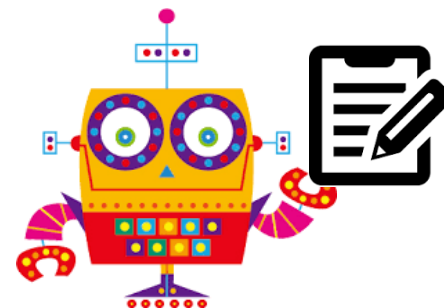


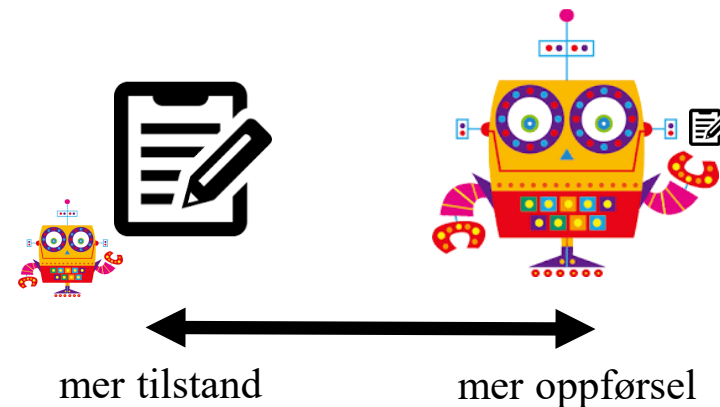
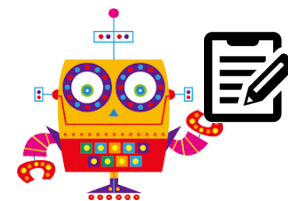
Bruk av objekter

- Objekter er en del av et *kjørende* program som har en
 - tilstand – data den husker
 - oppførsel – spørsmål du kan stille og oppgaver du kan be den utføre
- Objektene må lages (med **new**), ofte med en spesifikk start-tilstand
- Tilstanden til et objekt er ofte skjult, dvs. er ofte indirekte tilgjengelig gjennom lese-operasjoner
- Oppførselen er ofte spesifisert i [dokumentasjonen](#) og kan prøves ut ved å eksperimentere med objektene...



Tilstand og oppførsel-dualiteten

- Tilstanden er det et objekt husker
 - lagres i *attributter* (i java-terminologi kalles det *felt*)
 - tilstanden er enten et poeng i seg selv eller et middel for å realisere en viss oppførsel
- Oppførselen er det du kan be den om å gjøre
 - implementeres i operasjoner/metoder
 - (implisitte) regler for når og hvordan de kan kalles og hva de gjør
 - oppførselen er enten et poeng i seg selv eller et middel for å administrere tilstanden
- Et objekt med fokus på
 - *tilstand* er *data*-orientert
 - *oppførsel* er *tjeneste*-orientert



Flere strategier

- Bruker JShell for interaktiv utprøving

```

[dhcp-110-148:examples hal$ jshell
| Welcome to JShell -- Version 9.0.1
| For an introduction type: /help intro

[jshell> String s = "Java er gøy!"
s ==> "Java er gøy!"

[jshell> s.substring(8)
$2 ==> "gøy!"

[jshell> java.util.Random rand = new java.util.Random()
rand ==> java.util.Random@31dc339b

[jshell> rand.n
nextBoolean()    nextBytes(      nextDouble()    nextFloat()    nextGaussian()  nextInt(
[jshell> rand.nextInt(10)
$4 ==> 2

[jshell> rand.nextInt(10)
$5 ==> 0

jshell>

```

- (mens vi er her: to måter å lage tall: primitivt tall: *int i* vs objekter: *Integer i*)
- main-metode i klasse og JavaFX/FXML...

java.lang.String

- **String**-objekter kan lages på ulike måter
 - **String**-objekter kan legges rett inn i Java- programmer med “...”
 - “gjøre om” verdier og objekter med **String.valueOf(...)** og **obj.toString()**
 - bruke deler av eksisterende med **substring** (se under)
 - implisitt med +-operatoren, f.eks. “**Hall**” + “**vard**” gir “**Hallvard**”
 - basert på “mal”, f.eks. **String.format(“Navn: %s”, name)**
- **String**-objekter er en sekvens av tegn:
 - **charAt(pos)** – henter ut enkelt-tegn på bestemt posisjon
 - **substring(start)** – lager en ny **String** med alle tegnene fra **start** og utover
 - **substring(start, end)** – ny **String** med alle tegnene fra **start** og til **end**
 - **contains(s)** – sier om **s** finnes i dette **String**-objektet
 - **indexOf(s)** – returnerer posisjon til **s** i dette **String**-objektet, eller -1
 - **split(regex)** – deler opp ihht. **regex** for skilletegn

java.lang.String

- +-operatoren slår sammen **String**-objekter
 - “java “ + “er” + “ gøy” evalueres til “java er gøy”
 - operander som ikke **String**-objekter fra før, konverteres automatisk med **toString()** og **String.valueOf(...)**
- \ brukes foran spesielle bokstaver i String-konstanter
 - \” for å legge inn anførselstegn inni en tekst
 - \n for linjeskift (newline), \t for tabulator-tegnet
 - \u for Unicode-tegn generelt (prøv f.eks. \u2660)
- Merk at det ikke finnes metoder for å endre en **String**!
 - såkalt “immutable”, som for Python
- <https://www.ntnu.no/wiki/display/tdt4100/java.lang.String>

System.out

- Globale variabler brukt til input/output fra/til konsollet (tastaturet)
 - **System.out** – output som enten vises i konsollet eller i Console/Debug-panelet i editoren
 - **System.in** – input fra konsollet eller Console-panelet
- **System.out**
 - `println(...)` – skriver ut ... (hva som helst) med linjeskift etter
 - `print(...)` – skriver ut ... (hva som helst) uten linjeskift etter

System.in

Brukes ikke alene, men som inputkilde til `java.util.Scanner` eller `InputStreamReader` (mer om det senere)

- `Scanner scanner = new Scanner(System.in);`
- `String tekst = scanner.nextLine();`
- `System.out.println("Tekst: " + tekst);`
- `scanner.close`

java.util.Scanner

- Hjelpeklasse for innhenting og tolking av input
- Initialiseres med input-kilden, typisk System.in
- Venter på én og én linje input
- To typer metoder
 - hasNextXYZ() – sier om det ligger en bestemt type (ubrukt) input på vent
 - nextXYZ() – bruker (opp) input av en bestemt type
- Syntaks
 - hasNextBoolean()/nextBoolean() – true eller false
 - hasNextInt()/nextInt() – sekvens med siffer
 - hasNextDouble()/nextDouble() – sekvens med siffer og evt. desimalpunktum og flere siffer
 - hasNextLine ()/nextLine() – hel linje med tekst
 - hasNext()/next () - ett ord, dvs. bokstaver skilt av mellomrom og linjeskift
- <https://www.ntnu.no/wiki/display/tdt4100/java.util.Scanner>

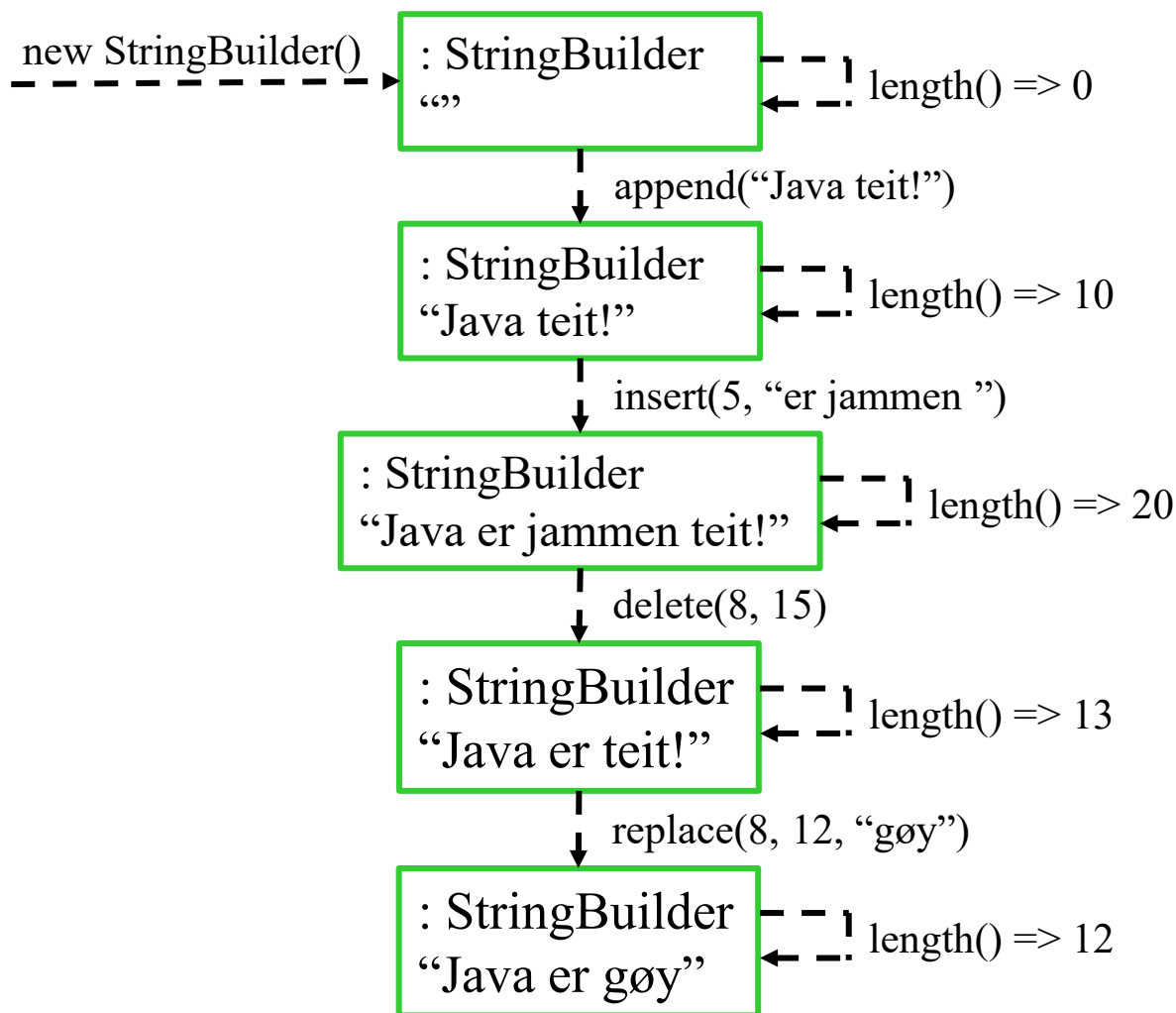
Testing med main-metode

- Et Java-program startes ved å aktivere en såkalt main-metode i en klasse:
 - **public static void main(String[] args) { ... }**
 - erstatt ... med oppstartskoden, f.eks. initialisering av objekter, kall av metoder osv. tilsvarende koden i en Scrapbook Page
 - Main-metoden i Klassen du har oppe kan kjøres på mange måter, avhengig av IDE og utvidelser.
- Merk at
 - main-metoden tar inn en **String**-tabell (array) som oppgis på kommandolinja eller i en egen “launch”-dialog
 - **static** betyr at **main**-metoden kjører utenfor et objekt, og virker sånn sett mer som en vanlig Python-funksjon
 - mest praktisk for rask testing av metodene i en klasse

java.lang.StringBuilder

- Objekt for å bygge opp større tekster, lages med
 - **new StringBuilder()** – uten innhold
 - **new StringBuilder(capacity)** – uten innhold, men forberedt for en viss kapasitet
 - **new StringBuilder(s)** – med s som initielt innhold
- I motsetning til **String**, så endres objektet selv:
 - **append(s)** – legger s til på slutten
 - **append(o)** – som over, men konverterer o til **String** først
 - **insert(pos, s)** – skyter s inn på den angitte posisjonen
 - **insert(pos, o)** – ... med implisitt konvertering
 - **delete(start, end)** – fjerner (trekker sammen) teksten fra og med **start** til **end**
 - **replace(start, end, s)** – erstatter teksten fra og med **start** til **end** med s
 - **toString()** – returnerer en **String** med tilsvarende innhold
- <https://www.ntnu.no/wiki/display/tdt4100/java.lang.StringBuilder+og+java.lang.StringBuffer>

StringBuilder-eksempel

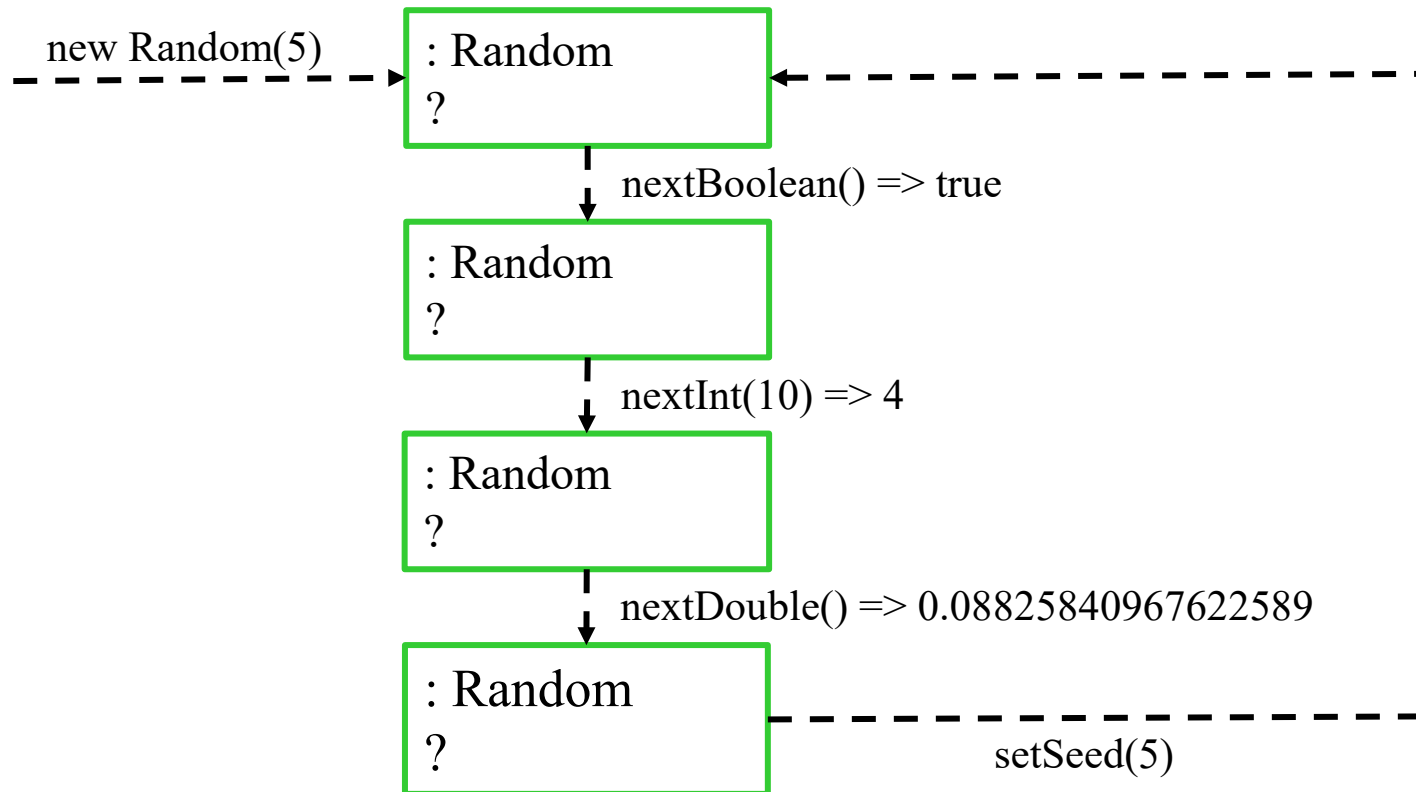


java.util.Random

- Objekt som genererer (pseudo)tilfeldige tall
- Fra en start-tilstand genereres en *fast* sekvens tall, som *virker* tilfeldig (tilfredsstiller visse fordelingskrav)
- Start-tilstanden kan settes eller være “tilfeldig”
 - **Random()** – “tilfeldig” start-tilstand
 - **Random(num)** – bestemt start-tilstand
- Ulike nextXYZ-metoder genererer neste verdi
 - **nextBoolean()** – genererer true eller false
 - **nextInt(max)** – genererer et heltall mellom 0 og opptil (men ikke med) max
 - **nextDouble()** – genererer et heltall mellom 0 og opptil (men ikke med) 1.0
 - **setSeed(num)** – setter (den interne) tilstanden (restarter sekvensen)

java.util.Random

(Objektilstandsdiagram)



java.util.List.of(...)

- Metode for å lage konstant-lister
 - `List<String> stringliste = List.of("1", "to", "III");`
- Typen elementer i argumentlista er viktig
 - Typen mellom `< >` bestemmer typen objekter som kan puttes inn og hentes ut
 - Må stemme med typen til elementene i argumentlista
 - Verdityper som **boolean**, **char**, **int** og **double** har egne objekttyper **Boolean**, **Character**, **Integer** og **Double** som må brukes...
- Metoder for å bruke innholdet
 - `size()` – returnerer antatt elementer
 - `get(pos)` – returnerer elementet på den angitt posisjonen
 - `contains(o)/indexOf(o)` – sier om o finnes i lista/hvor i lista o finnes (eller -1)
- Kan gå gjennom lista med **for**:

```
for (String s : List.of("1", "to", "III")) {
    System.out.println(s);
}
```

java.util.ArrayList

- Generelt objekt for å håndtere lister med objekter, inkludert lista (i motsetning ved bruk av **List.of(...)**)
- Når lista lages eller deklarereres, så må en oppgi typen elementer en har tenkt å legge inn
 - `ArrayList<String> stringliste = new ArrayList<String>();`
 - Typen mellom `< >` bestemmer typen objekter som kan puttes inn og hentes ut
- Metoder:
 - `size()` – returnerer antatt elementer
 - `get(pos)` – returnerer elementet på den angitt posisjonen
 - `set(pos, o)` – bytter ut elementet på den angitte posisjonen med `o`
 - `add(o)/remove(o)` – legger til `o` bakerst i lista/fjerner `o`
 - `add(pos, o)/remove(pos)` – skyter `o` inn på den angitte posisjonen/fjerner element
 - `contains(o)/indexOf(o)` – sier om `o` finnes i lista/hvor i lista `o` finnes (eller -1)
 - `toString()` – returnerer en **String** med innhold mellom `[]`
- <https://www.ntnu.no/wiki/display/tdt4100/java.util.ArrayList>

ArrayList-eksempel

(Objekttilstandsdiagram)

