

Pakker og innkapsling

INF101 forelesning 7. Februar 2022

Torstein Strømme

men først litt repitisjon

Repetisjon: Hva skjer her, egentlig?

```
public class App {
    static String DEFAULT_TEXT = "Hello World";
    String text;

    public static void main(String[] args) {
        String text = DEFAULT_TEXT;
        if (args.length > 0) {
            text = args[0];
        }
        App app = new App(text);
        app.run();
    }

    App(String text) {
        this.text = text;
    }

    void run() {
        System.out.println(this.text);
    }
}
```

På kommandolinjen:
\$ javac App.java
\$ java App Hipp hurra

Repetisjon: Hva skjer her, egentlig?

```
public class App {  
    static String DEFAULT_TEXT = "Hello World";  
    String text;  
  
    public static void main(String[] args) {  
        String text = DEFAULT_TEXT;  
        if (args.length > 0) {  
            text = args[0];  
        }  
        App app = new App(text);  
        app.run();  
    }  
  
    App(String text) {  
        this.text = text;  
    }  
  
    void run() {  
        System.out.println(this.text);  
    }  
}
```

String-objekt

Kompilert byte-kode

11

'H'

'e'

'l'

'l'

'o'

' '

'W'

'o'

'r'

'l'

'd'

XXXX

XXXX

...

...

XXXX

XXXX

Repetisjon: Hva skjer her, egentlig?

DEFAULT_TEXT

```
public class App {
    static String DEFAULT_TEXT = "Hello World";
    String text;

    public static void main(String[] args) {
        String text = DEFAULT_TEXT;
        if (args.length > 0) {
            text = args[0];
        }
        App app = new App(text);
        app.run();
    }

    App(String text) {
        this.text = text;
    }

    void run() {
        System.out.println(this.text);
    }
}
```

null
11
'H'
'e'
'l'
'l'
'o'
''
'W'
'o'
'r'
'l'
'd'
XXXX
XXXX
...
...
XXXX
XXXX

Repetisjon: Hva skjer her, egentlig?

```
public class App {
    static String DEFAULT_TEXT = "Hello World";
    String text;

    public static void main(String[] args) {
        String text = DEFAULT_TEXT;
        if (args.length > 0) {
            text = args[0];
        }
        App app = new App(text);
        app.run();
    }

    App(String text) {
        this.text = text;
    }

    void run() {
        System.out.println(this.text);
    }
}
```

DEFAULT_TEXT

11
'H'
'e'
'l'
'l'
'o'
' '
'W'
'o'
'r'
'l'
'd'
XXXX
XXXX
...
...
XXXX
XXXX

Repetisjon: Hva skjer her, egentlig?

```
public class App {
    static String DEFAULT_TEXT = "Hello World";
    String text;

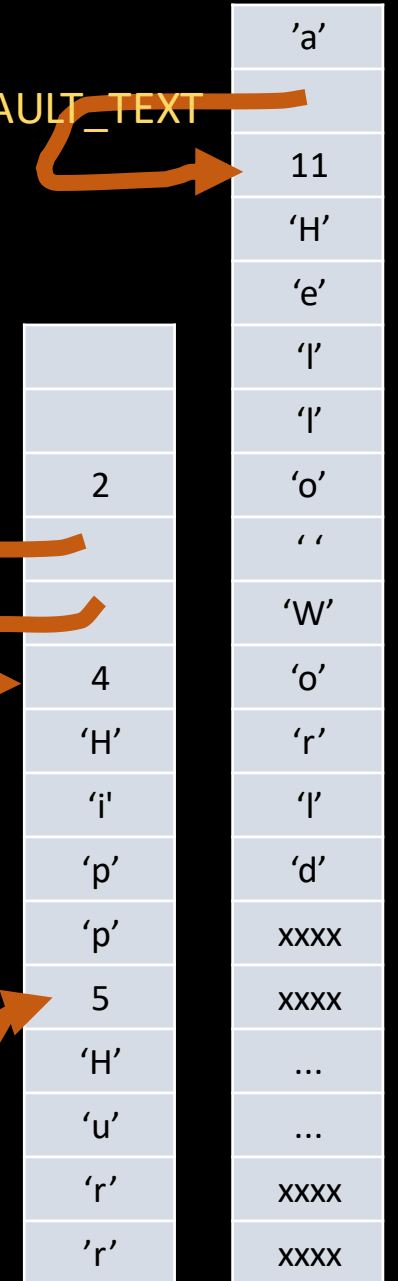
    public static void main(String[] args) {
        String text = DEFAULT_TEXT;
        if (args.length > 0) {
            text = args[0];
        }
        App app = new App(text);
        app.run();
    }

    App(String text) {
        this.text = text;
    }

    void run() {
        System.out.println(this.text);
    }
}
```

Array med strenger

DEFAULT_TEXT



På kommandolinjen:
\$ javac App.java
\$ java App Hipp hurra

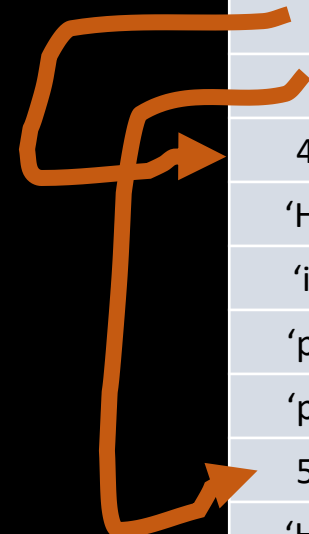
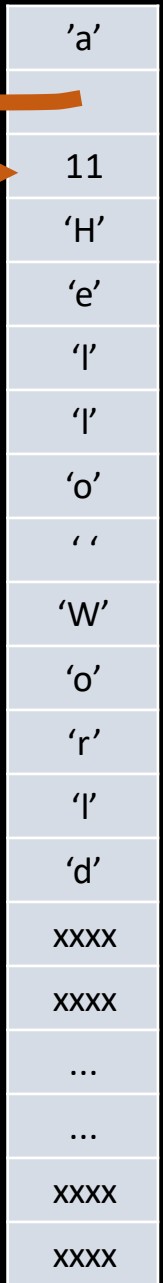
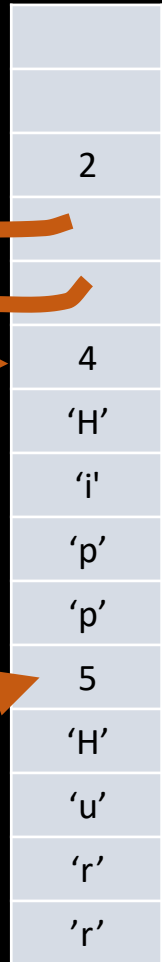
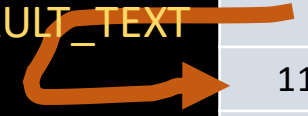
Repetisjon: Hva skjer her, egentlig?

stack

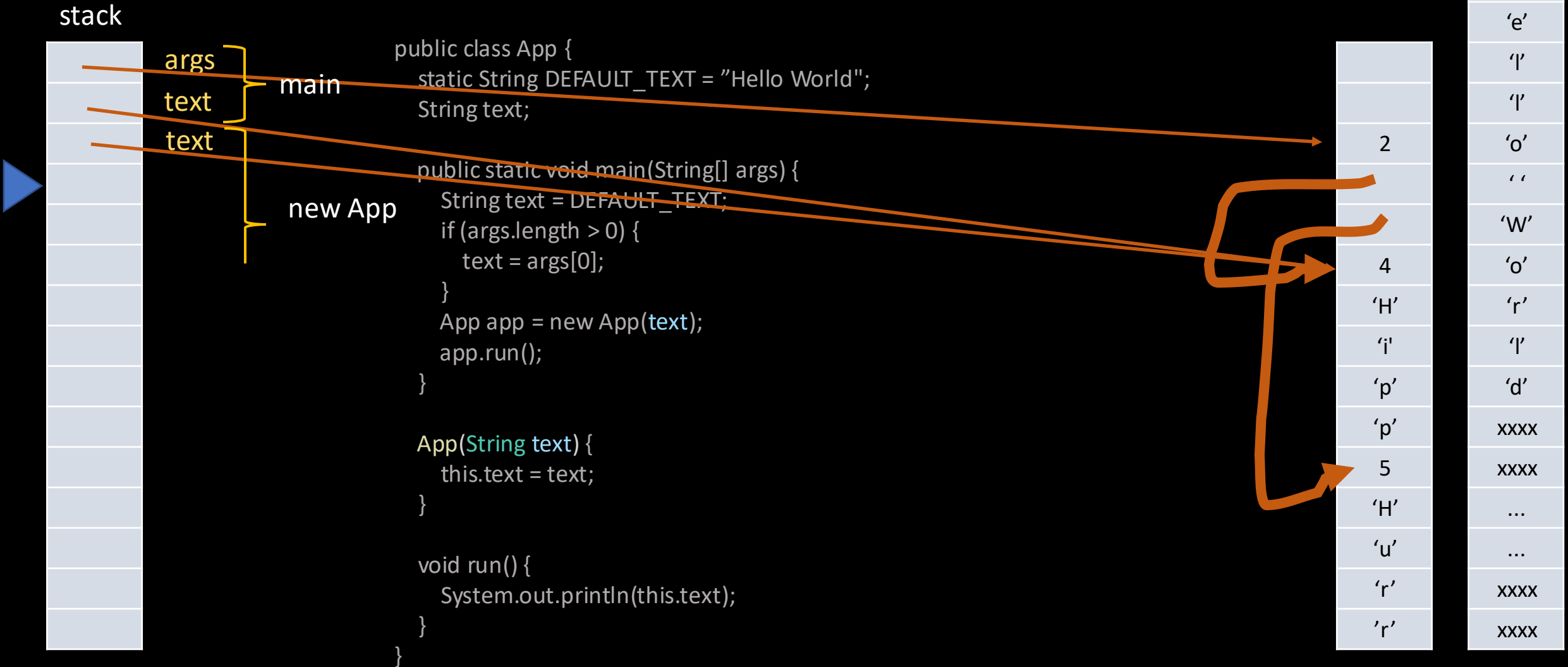
args

```
public class App {  
    static String DEFAULT_TEXT = "Hello World";  
    String text;  
  
    public static void main(String[] args) {  
        String text = DEFAULT_TEXT;  
        if (args.length > 0) {  
            text = args[0];  
        }  
        App app = new App(text);  
        app.run();  
    }  
  
    App(String text) {  
        this.text = text;  
    }  
  
    void run() {  
        System.out.println(this.text);  
    }  
}
```

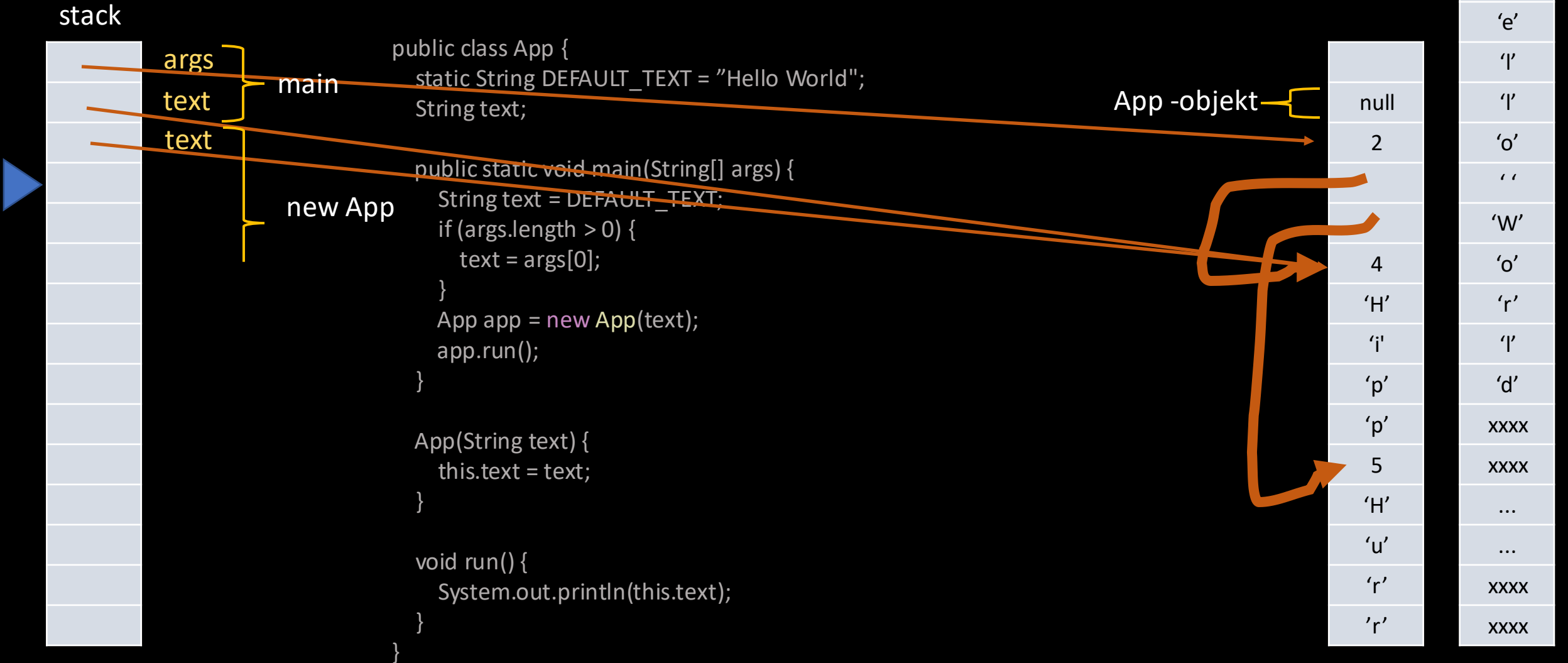
DEFAULT_TEXT



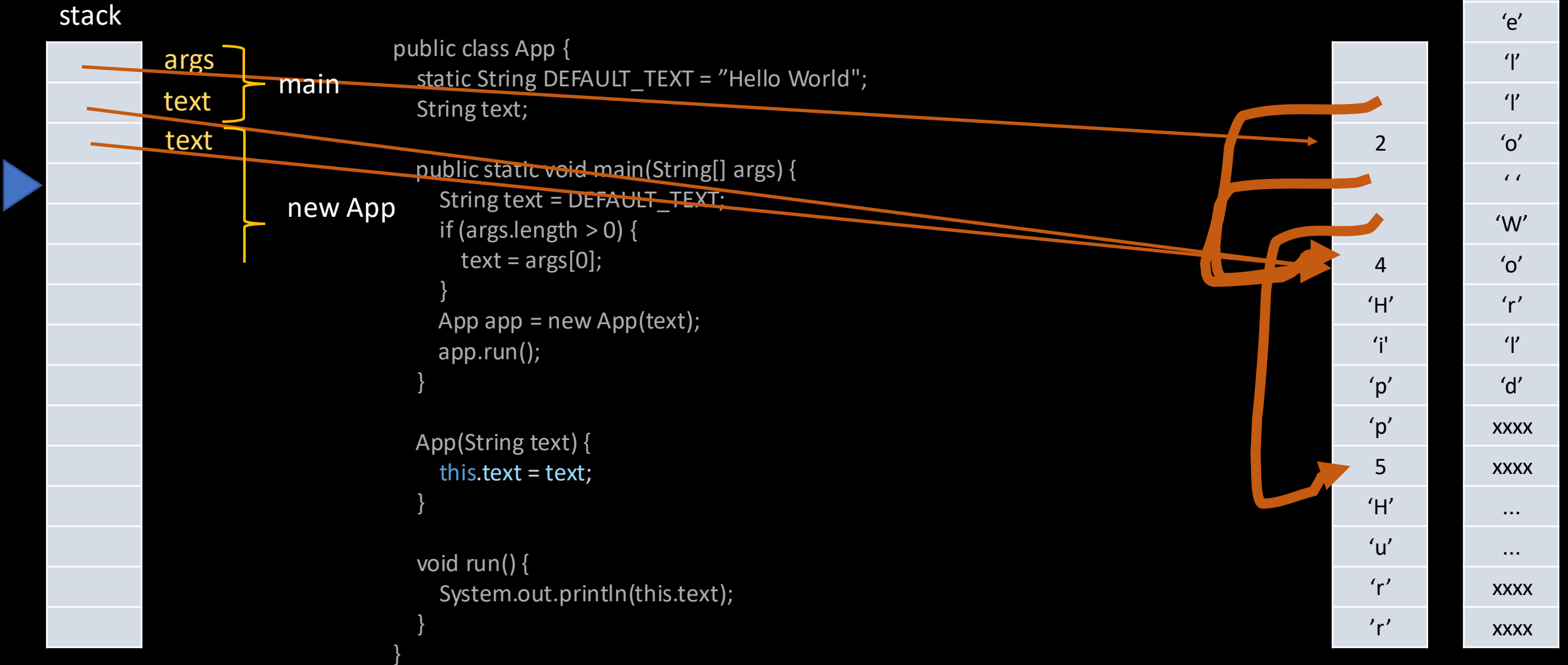
Repetisjon: Hva skjer her, egentlig?



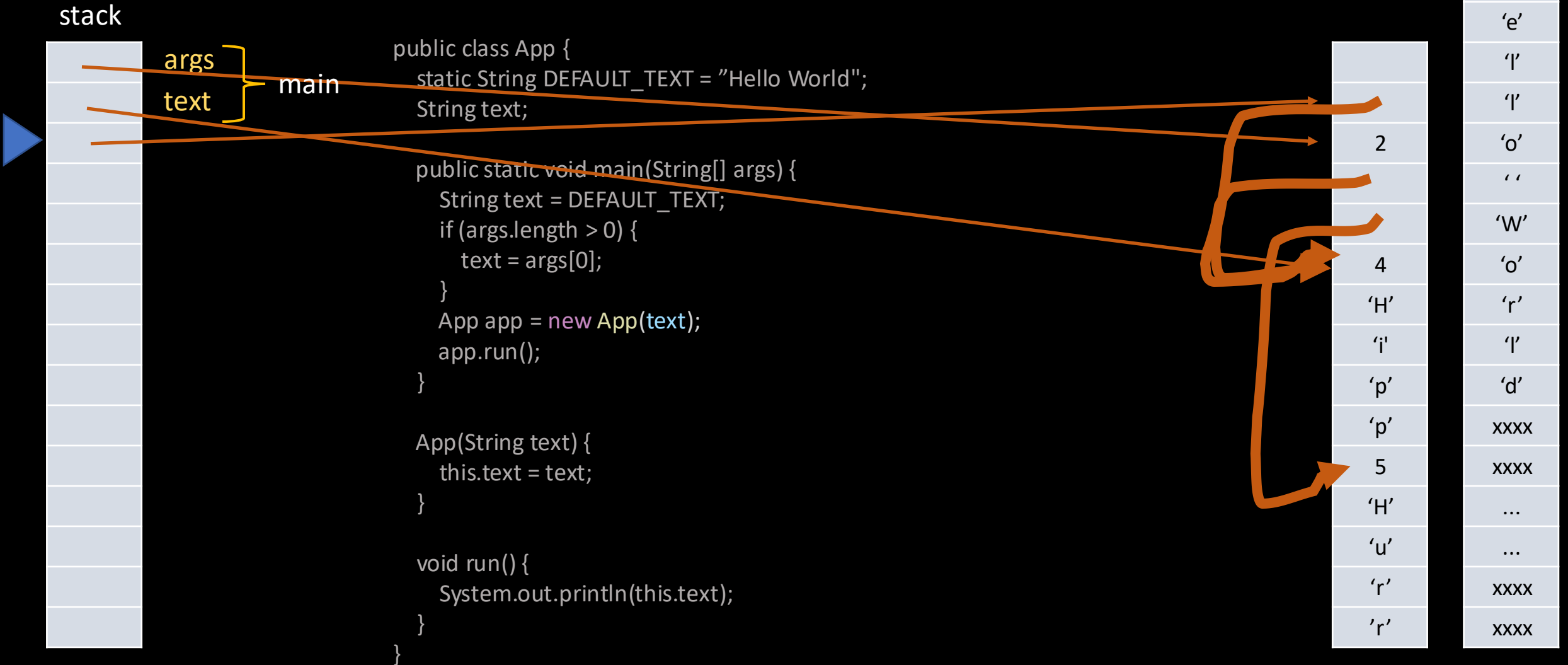
Repetisjon: Hva skjer her, egentlig?



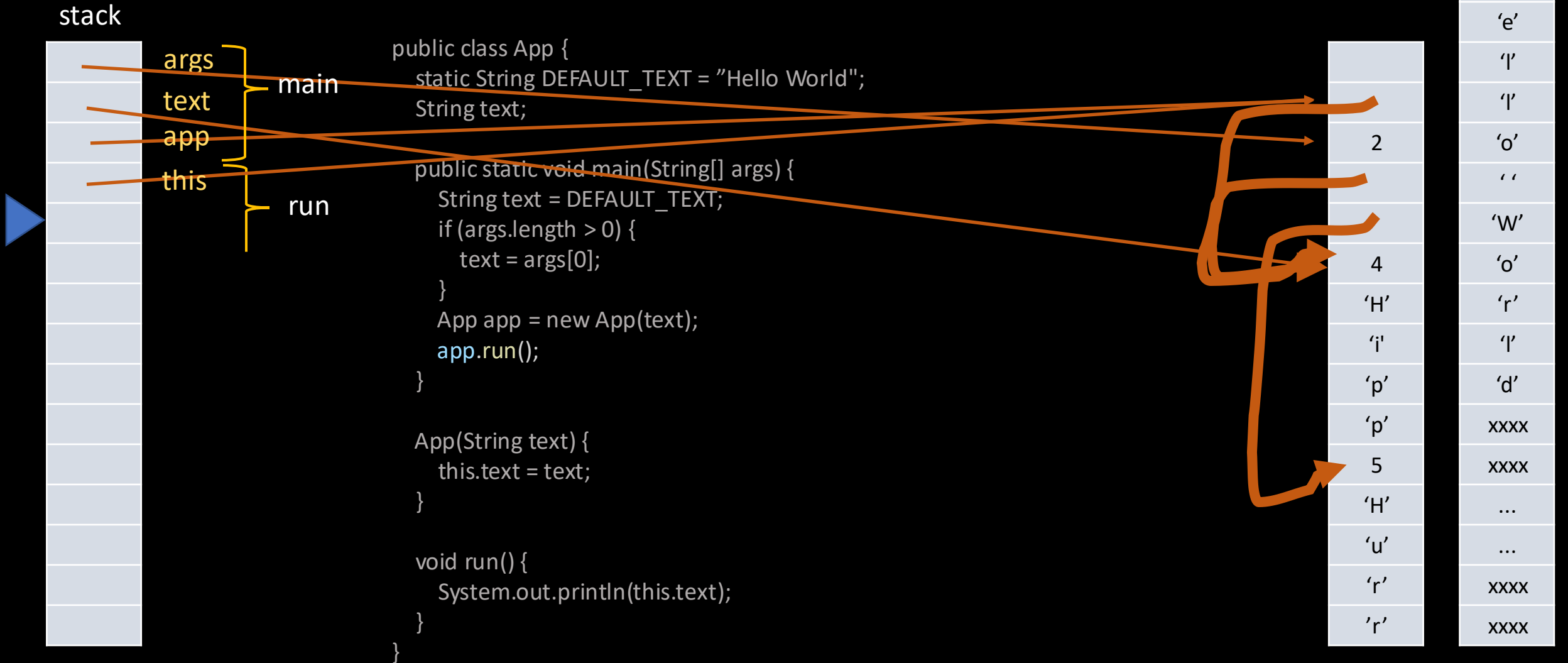
Repetisjon: Hva skjer her, egentlig?



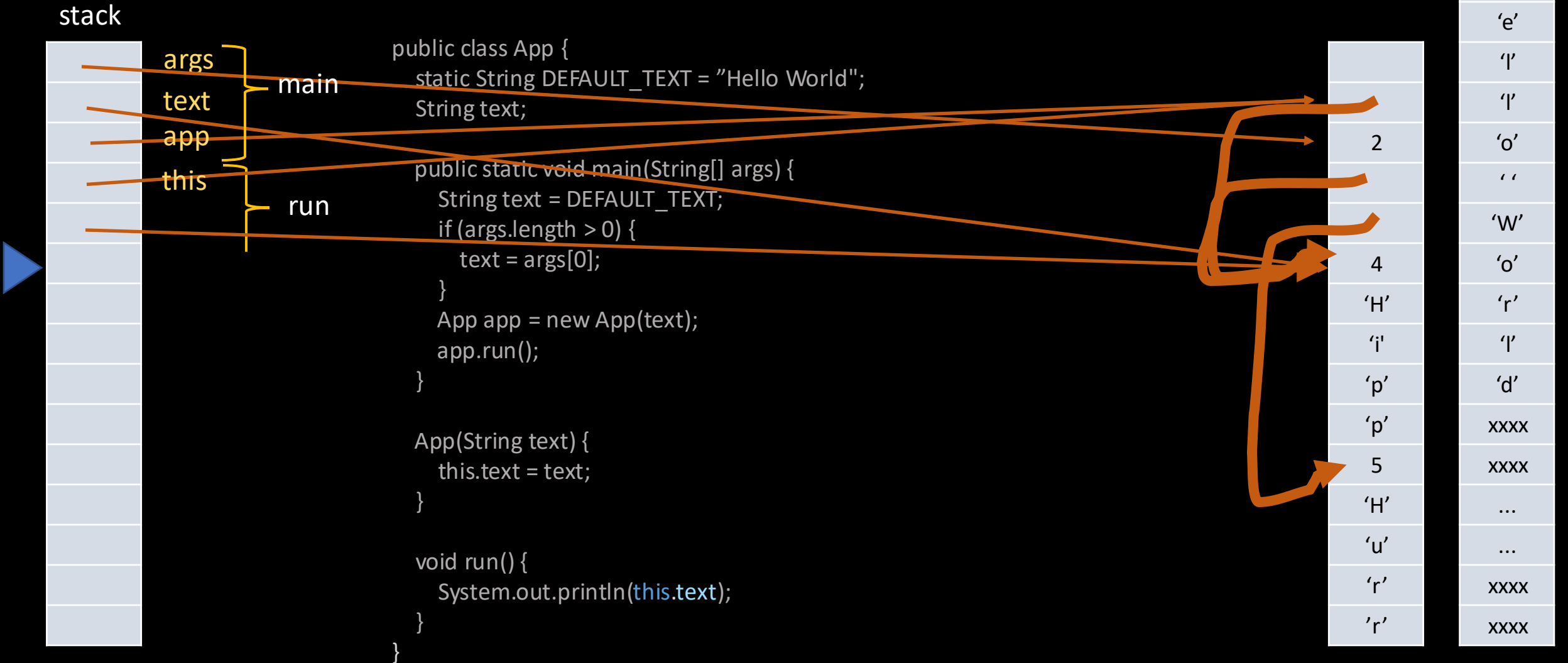
Repetisjon: Hva skjer her, egentlig?



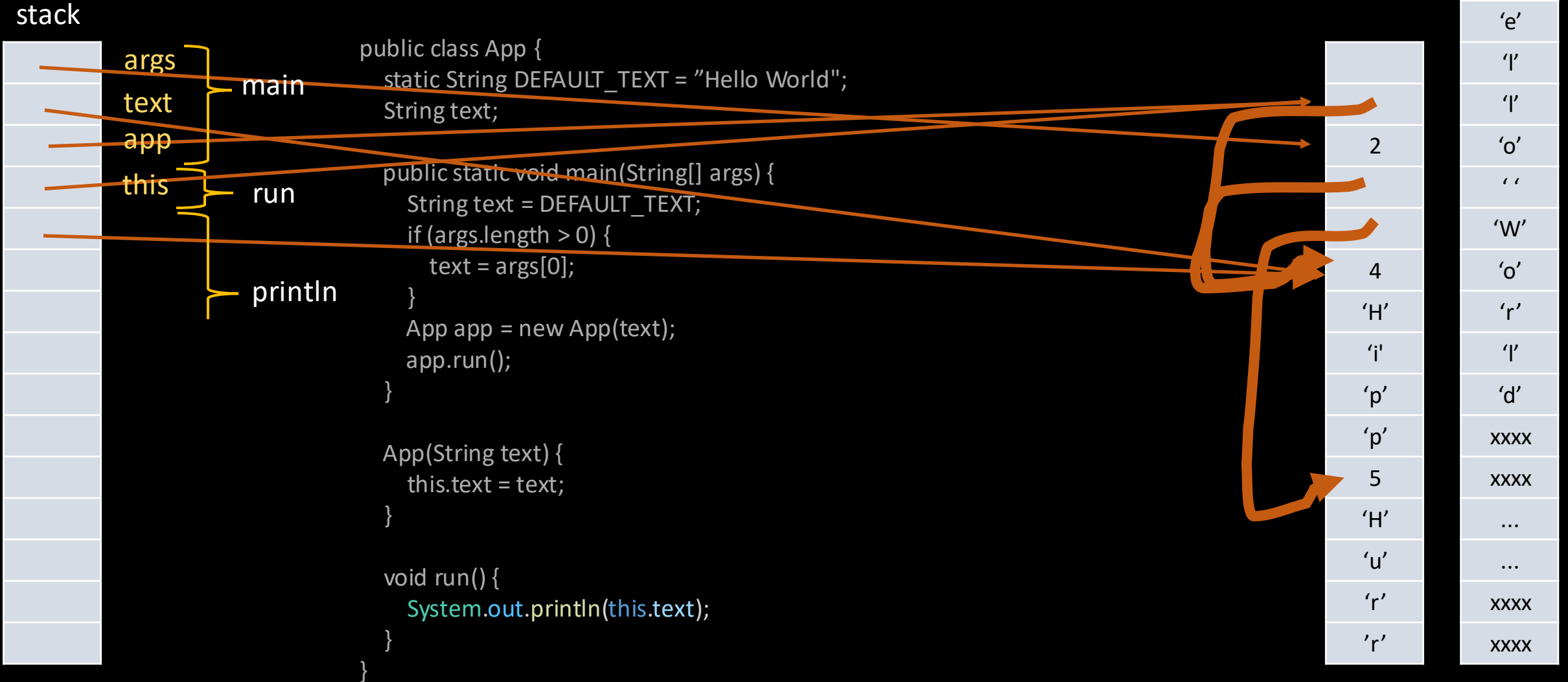
Repetisjon: Hva skjer her, egentlig?



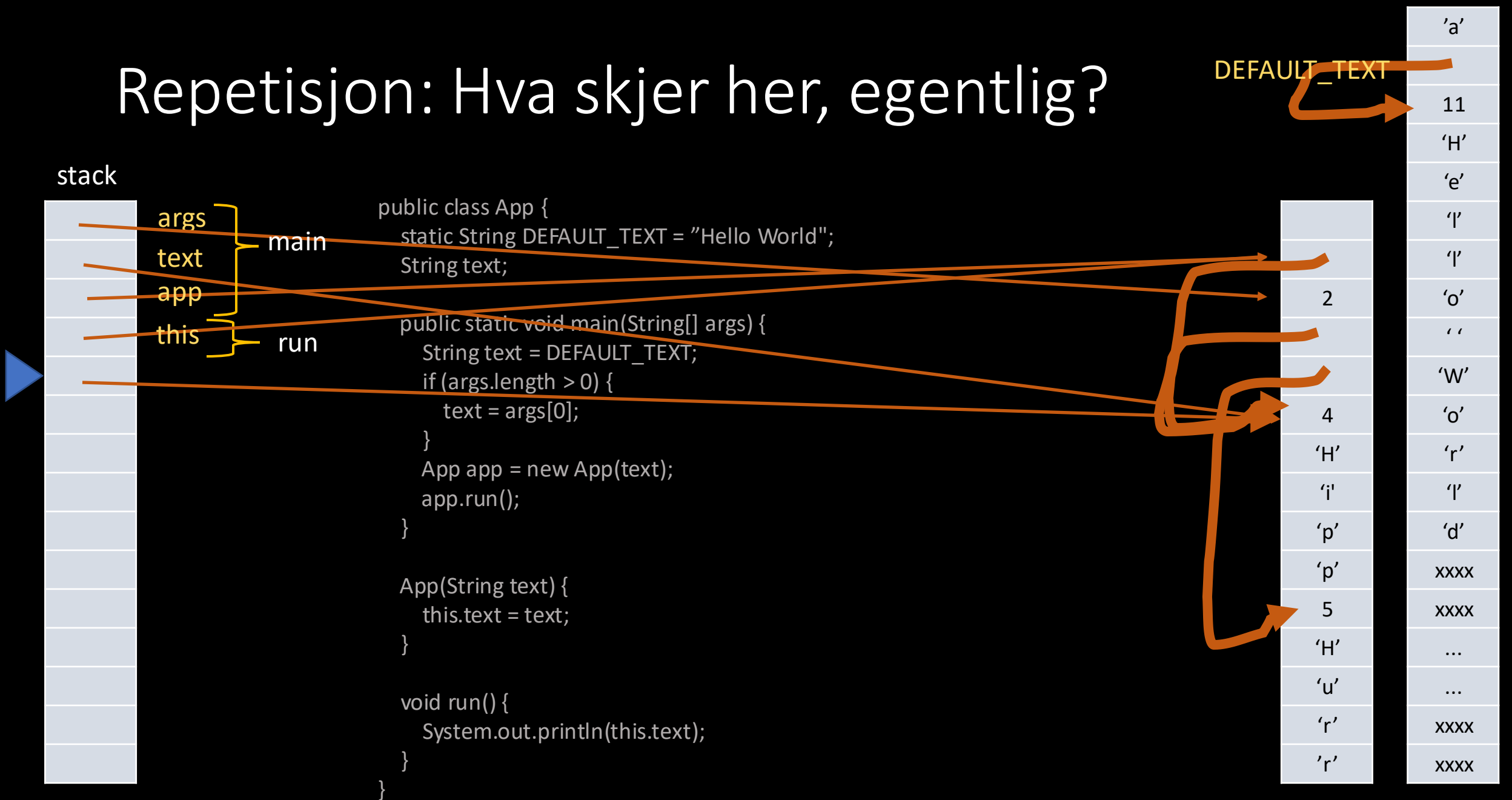
Repetisjon: Hva skjer her, egentlig?



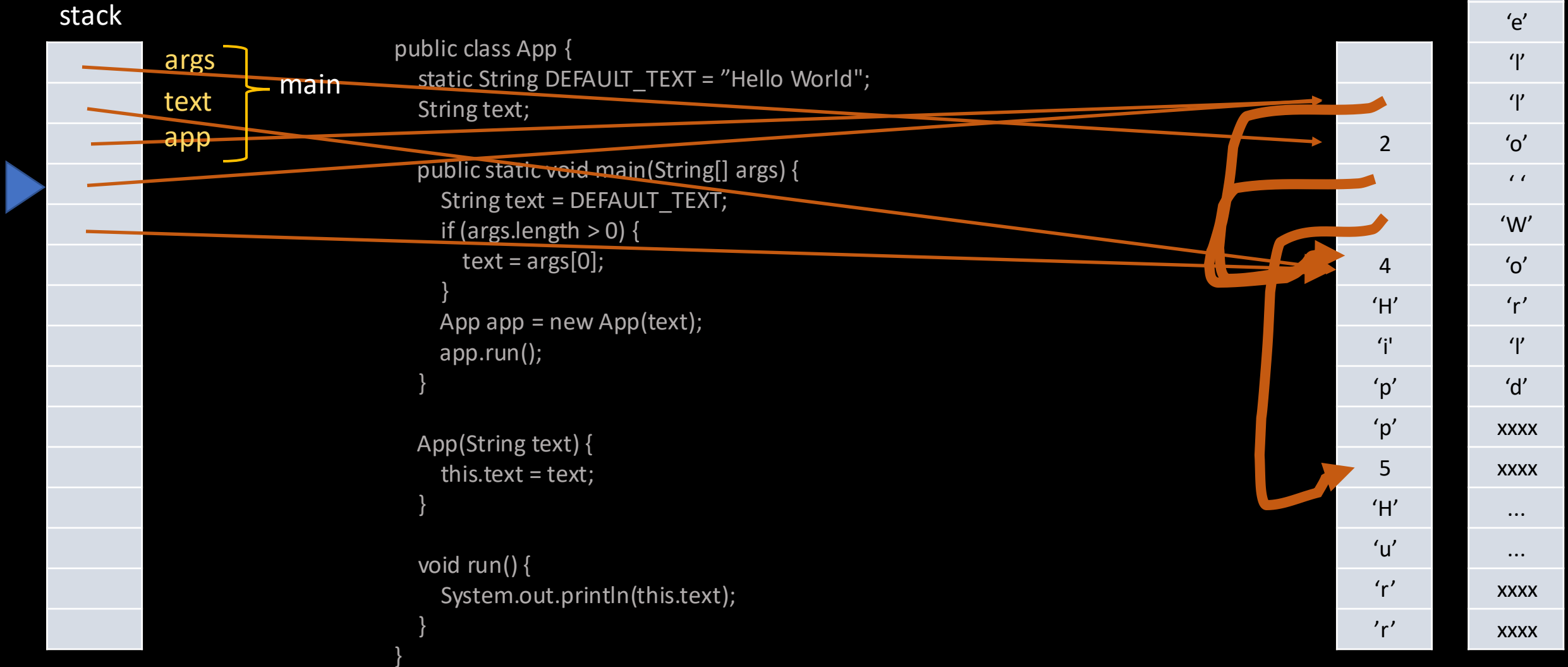
Repetisjon: Hva skjer her, egentlig?



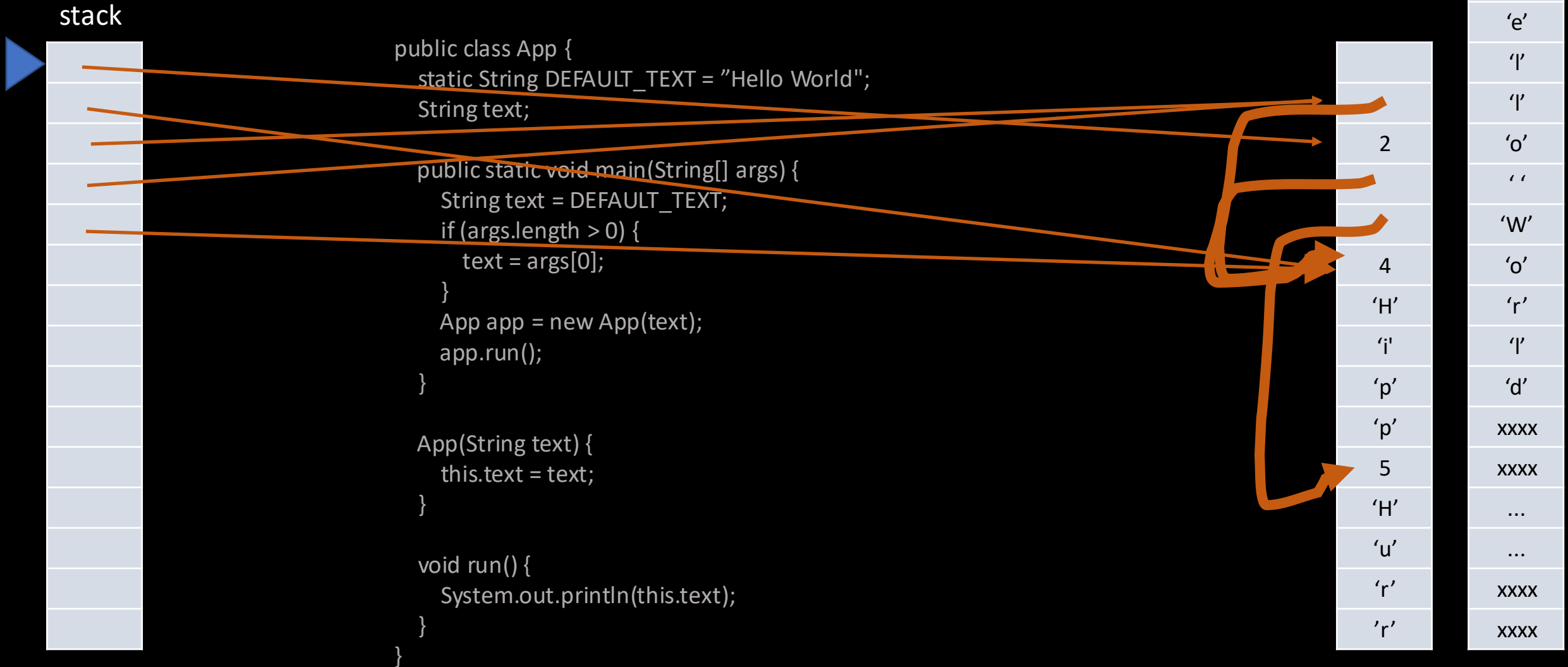
Repetisjon: Hva skjer her, egentlig?



Repetisjon: Hva skjer her, egentlig?



Repetisjon: Hva skjer her, egentlig?



Repetisjon: kodeord static

Tilhører klassen

Tilhører objekter

Statiske metoder

- Velegnet til *funksjoner* (metoder som ikke har side-effekter, kun retur-verdier)

```
Klasse.metodeNavn();
```

Statiske variabler

- Globale variabler, tilgjengelig for “alle”
- Velegnet til konstanter
- Bør ellers unngås
- Vanskelig å teste med

Instans-metoder

- Kan se og endre instans-variabler (har alltid tilgang til objektet *this*)

```
objekt.metodeNavn();
```

Instans-variabler

- Variabler som er unike for hvert objekt
- Velegnet for å holde på tilstander/state som logisk hører sammen.

Objektorientert programmering



Abstraksjon

- Hvilke egenskaper er relevante?



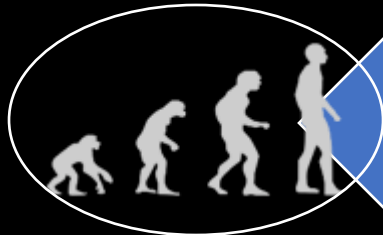
Modularitet

- Lag kode som byggeklosser



Innkapsling

- Skjul detaljer; ha kontroll over egne strukturer.



Gjenbruk

- DRY – don't repeat yourself
- Biblioteker, polymorfisme og generiske typer

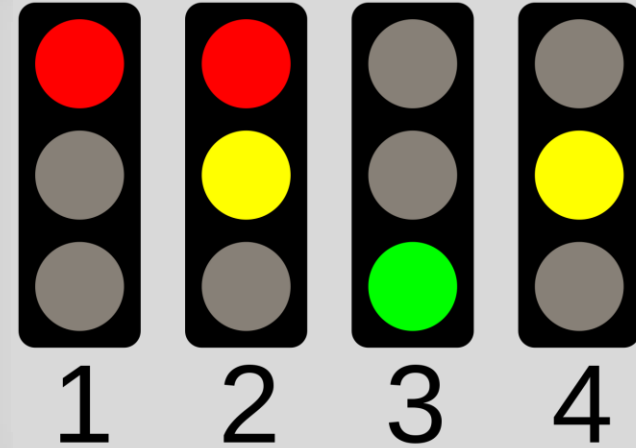
Eksempel: Trafikklys

- Vi skal implementere en klasse for et lyskryss
- Første steg: en klasse for ett trafikklys.



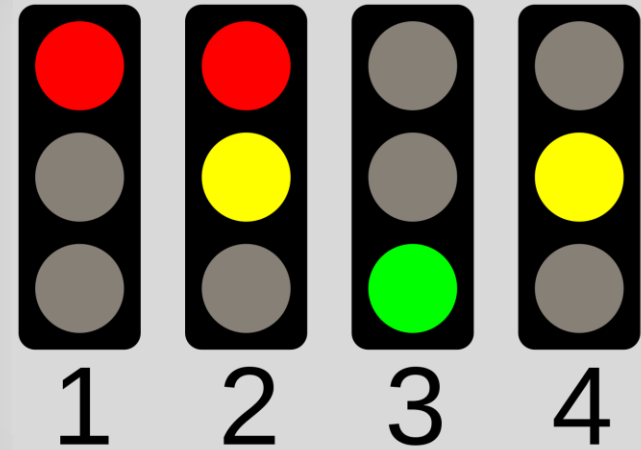
Eksempel: Trafikklys

- Første steg: abstraksjon
 - Hvordan representerer vi et trafikklys?
 - 3D –modell?
 - Fotografi?
 - Kretskortdesign?
 - True/false –verdier for de tre fargene?
 - Tall 1,2,3,4 som representerer tilstand?



Eksempel: Trafikklys

- Livekoding



Pakker

- Første linje i en .java –fil. For eksempel:

```
package inf101v22.trafficlight;
```

- Ordner klasser i logiske grupper
- Tillater at samme klassenavn benyttes flere ganger
- Tillater import av flere klasser på en gang. For eksempel:

```
import java.util.*;
```

Pakker

- I Maven-prosjekter matcher pakkenavn hvilken mappe filen ligger i. For eksempel vil en .java-fil med pakke

```
package inf101v22.trafficlight;
```

ligger i mappen `./src/main/java/inf101v22/trafficlight`

- Pakker skal kun ha lowercase bokstaver
- Brukes også som tilgangskontroll

Tilgangskontroll

- Hvilke klasser har tilgang til variabler og metoder?
- Ved å spesifisere enten
 - private
 - public
 - protected
 - (default)

vil metoder og variabler være tilgjengelig fra andre klasser i ulik grad

Tilgangsmodifikatorer (access modifiers)

Hvilke klasser har tilgang til variabelen/metoden?

	public	protected	default	private
Samme klasse	Ja	Ja	Ja	Ja
Samme pakke	Ja	Ja	Ja	Nei
Klasse som arver	Ja	Ja	Nei	Nei
Alle	Ja	Nei	Nei	Nei

Hvorfor skjule?

- “Low coupling”
- Modularitet
- Testbarhet
- Skiller ansvarsområder – “single-responsibility principle”
- Hindrer andre i å klusse til dine strukturer uten å endre på dine filer

Hvordan skjule?

- Grensesnitt
 - Hovedregel: alltid benytt et grensesnitt som type for variabler
 - Benytt det mest restriktive grensesnittet som lar deg utføre oppgaven du har ansvar for
- Tilgangsmodifikatorer
 - Benytt private som utgangspunkt for både variabler og metoder
 - Utvid tilgang gradvis ved behov; men vær sikker på at det er nødvendig.
 - Eksponer heller en metode enn en variabel