Adobe

Java DevCamp 2023

Girls edition

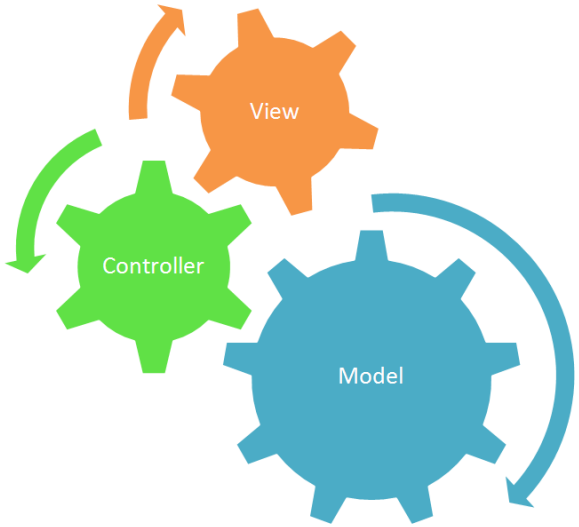
Agenda Workshop #3

Recap

Streams

Docker

Workshop #2 Recap



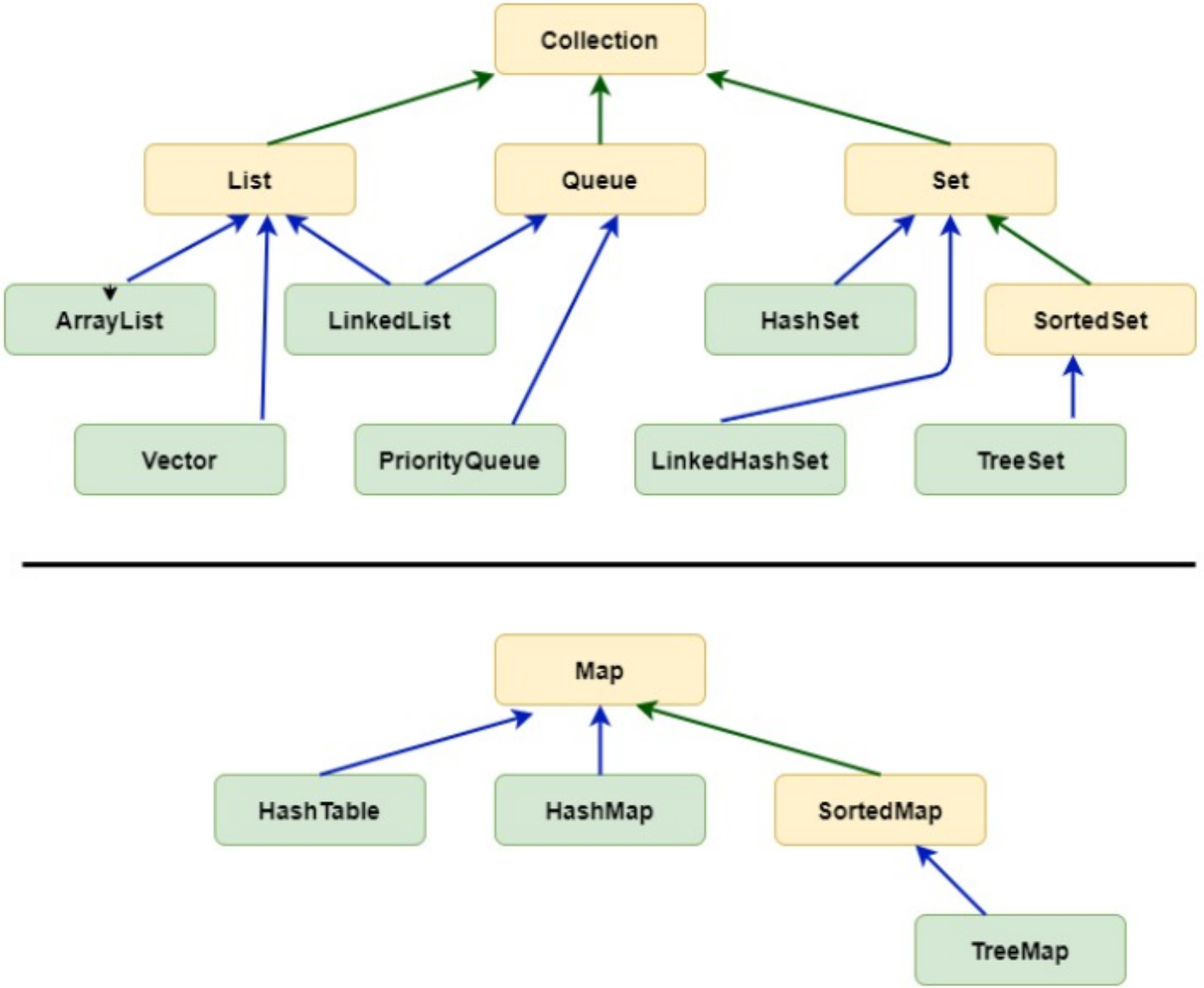
Maven™

JACKSON

Get the latest updates from the Git repo

Streams 🌼

Collections



Streams

Sequence of elements which allow functional and declarative operations on collections

Functional Programming Concepts

- Functions as first-class objects
  - F = +
  - Lambdas 🎀
    - (a, b) -> a + b
- Pure functions = member variables + changing state
- Higher-order functions = take one function as a parameter

Streams vs. Collections

- Conceptual
- Data modification
- Iteration
- Traversal
- Construction

Streams vs. Collections

- **Conceptual** - Collections are meant to store data, streams are meant to apply operations
- Data modification
- Iteration
- Traversal
- Construction

Streams vs. Collections

- Conceptual
- **Data modification** – A stream consumes a view and returns a result, without altering it
- Iteration
- Traversal
- Construction

Streams vs. Collections

- Conceptual
- Data modification
- **Iteration** – On streams, iteration is done internally, depending on the chosen operations
- Traversal
- Construction

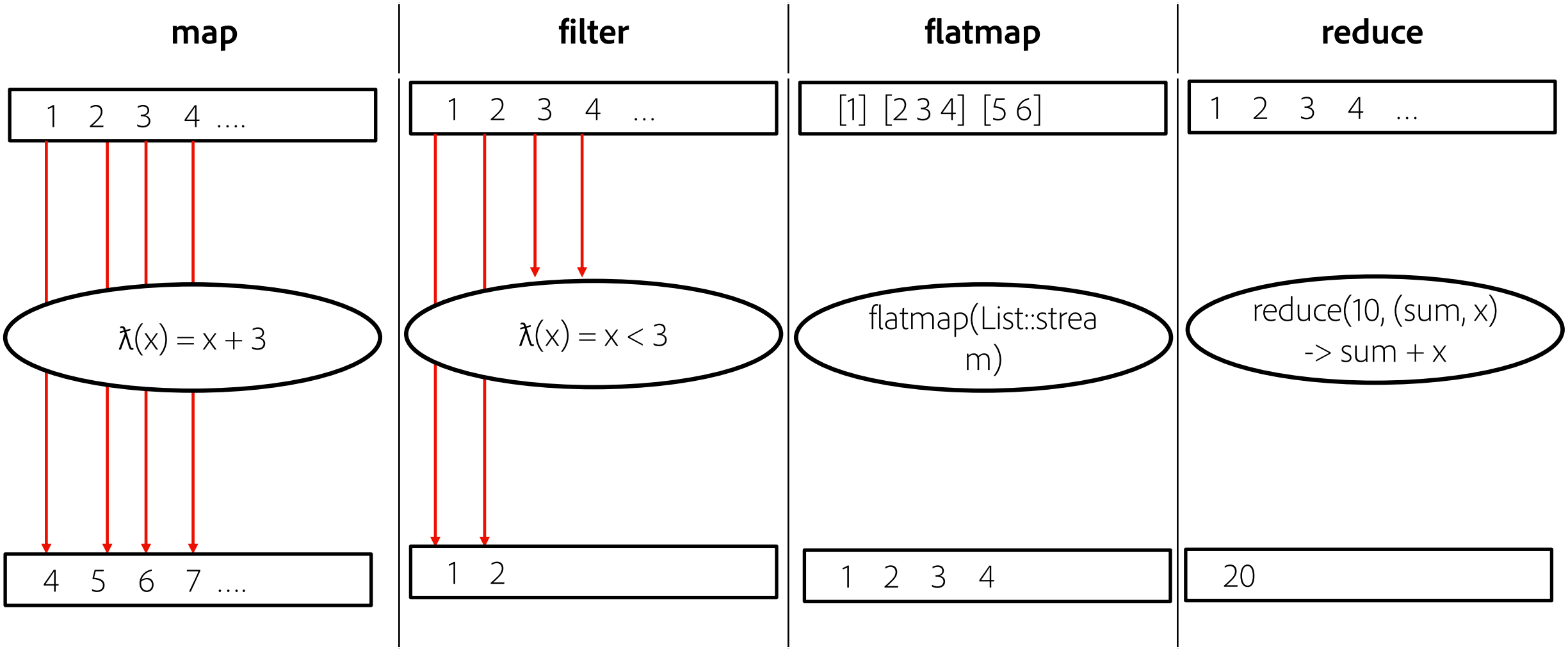
Streams vs. Collections

- Conceptual
- Data modification
- Iteration
- **Traversal** – Streams can be traversed only once
- Construction

Streams vs. Collections

- Conceptual
- Data modification
- Iteration
- Traversal
- **Construction** – Collections are eager, streams are lazy

Streams in action



Let’s bake some streams 👩‍🍳

Let’s install Docker 

Verify installation:

- docker --version

Dockerfile

- Set of instructions for Docker to build the image **layer by layer**
  - each instruction results in a new layer
  - layers are stacked on top of each other forming the final image

- **Caching** for build process
  - Unchanged instructions => layer has not changed => reuse the cached layer

Layer invalidation – initial Dockerfile

```
1  # Dockerfile
2
3  # Base image
4  FROM ubuntu:20.04
5
6  # Layer 1 - Installing initial dependencies
7  RUN apt-get update && apt-get install -y \
8      curl \
9      && rm -rf /var/lib/apt/lists/*
10
11 # Layer 2 - Adding a line to install additional dependencies
12 RUN apt-get update && apt-get install -y \
13     wget \
14     && rm -rf /var/lib/apt/lists/*
15
16 # Layer 3 - Copying application code
17 COPY app /app
18
```

Layer invalidation – modified Dockerfile

```
1  # Dockerfile
2
3  # Base image
4  FROM ubuntu:20.04
5
6  # Layer 1 - Modifying the line to install additional dependencies
7  RUN apt-get update && apt-get install -y \
8      curl \
9      vim \   # Modified line: added 'vim' package
10     && rm -rf /var/lib/apt/lists/*
11
12 # Layer 2 - Invalidated due to the modification in Layer 1
13 RUN apt-get update && apt-get install -y \
14     wget \
15     && rm -rf /var/lib/apt/lists/*
16
17 # Layer 3 - Invalidated due to the modification in Layer 1
18 COPY app /app
```

Addition of 'vim' =>
Layer 1 invalidated

+ subsequent layers,
Layer 2 and Layer 3
are also invalidated

Let's play with images & containers - Dockerfile

- List all images
  - **docker images**
- Build the docker image
  - **docker build -t <your-image-name>:<tag> .**
    - e.g docker build -t mysql:1.0.0 .
- Run a container from an image
  - **docker run <your-image-name>:<tag>**
- List running containers
  - **docker ps**

Let's play with images & containers – docker-compose

- Start services
  - **docker compose up**

- Stop & remove the containers (+ networks, volumes)
  - **docker compose down**


Adobe
Artwork by Frank Moth