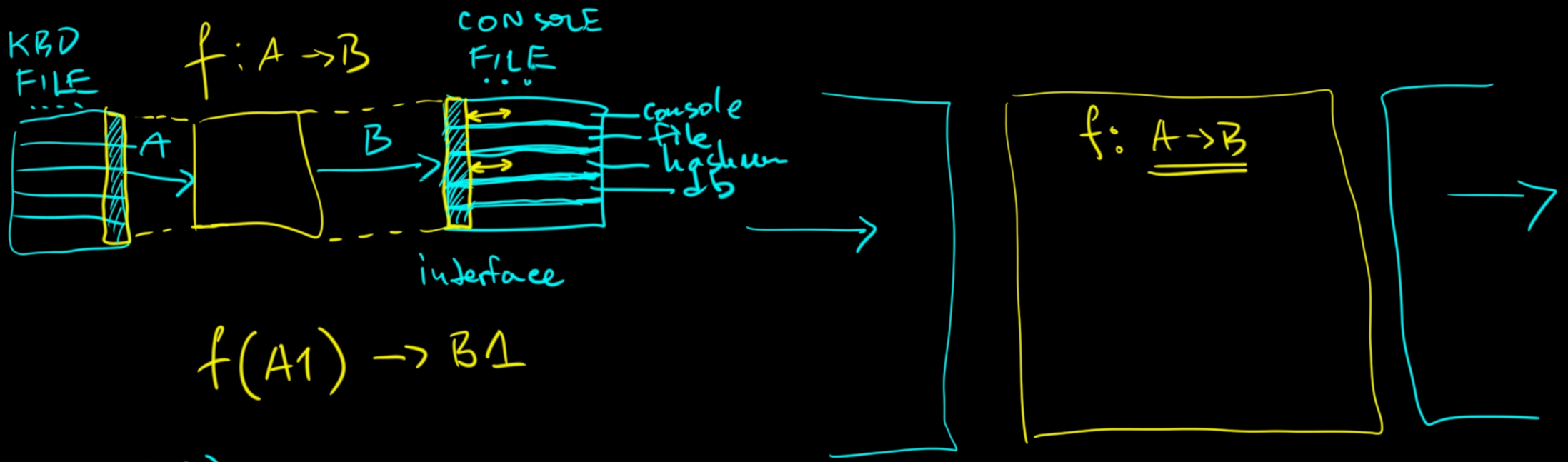
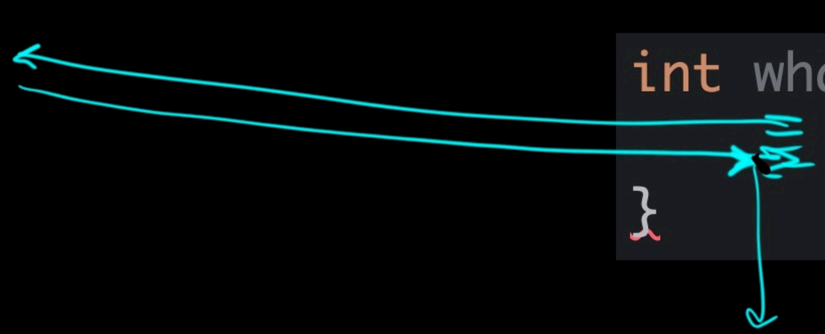


f(A1) -> B1

readFile(x) -> String

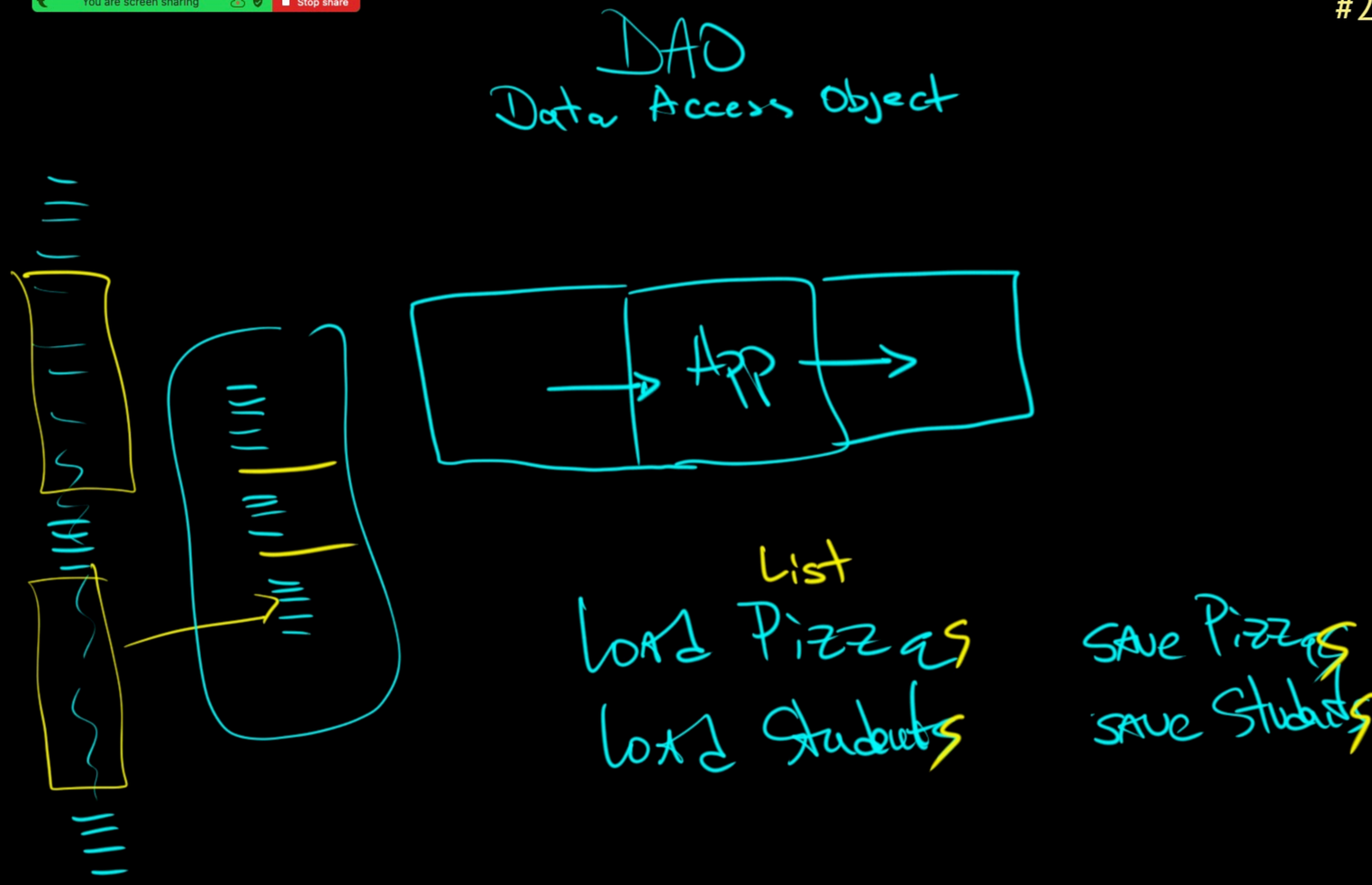
f:() -> A

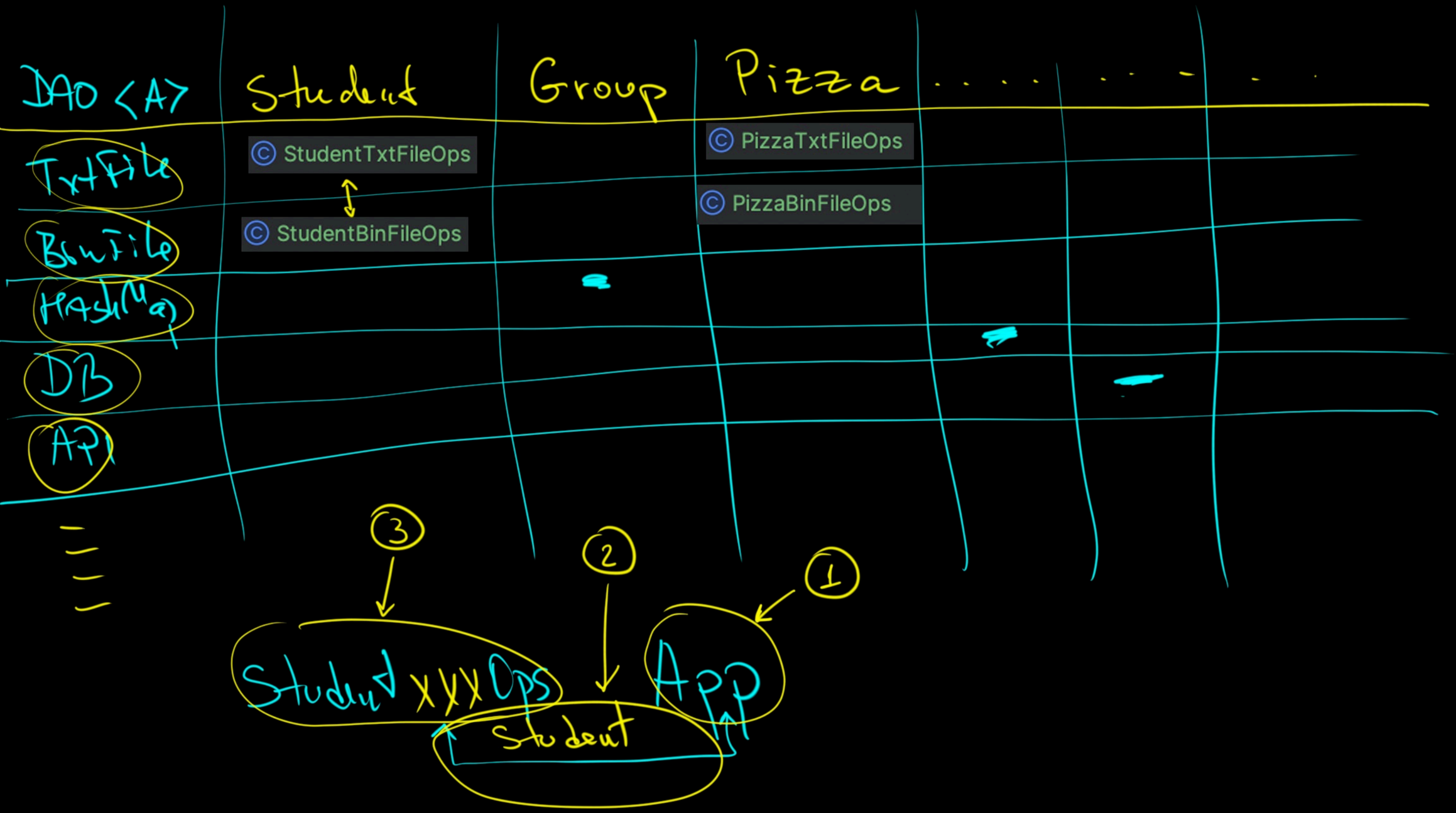


```
int whatever() {

}
```

```
void whatever2(int x) {

}
```

```java
static void save() throws IOException {
  Pizza p1 = new Pizza( name: "M1",  size: 30);
  Pizza p2 = new Pizza( name: "M2",  size: 35);

  BufferedWriter w = new BufferedWriter(new FileWriter(

  w.write(p1.represent());
  w.newLine();
  w.write(p2.represent());
  w.newLine();

  w.close();
}

static List<Pizza> load() throws IOException {
  BufferedReader r = new BufferedReader(new FileReader(
  List<Pizza> ps = r.lines() Stream<String>
    .map(Pizza::fromString) Stream<Pizza>
    .toList();
  r.close();
  return ps;
}

public static void main(String[] args) throws IOExcepti
//    save();
  List<Pizza> load = load();
  load.forEach(System.out::println);
```





```java
package lesson17;

import ...

public class StudentBinFileOps {

  private final String fileName;

  public StudentBinFileOps(String fileName) {
    this.fileName = fileName;
  }

  void save(List<Student> xs) throws IOException {
    try (var os = new ObjectOutputStream(new FileOutputS
      os.writeObject(xs);
    }
  }

  List<Student> load() throws IOException, ClassNotFound
    try(var is = new ObjectInputStream(new FileInputStre
      Object o = is.readObject();
      return (List<Student>) o;
    }
  }

}
```

```java
package lesson17;

import ...

public class StudentTxtFileOps {

  private final String fileName;

  public StudentTxtFileOps(String fileName) {
    this.fileName = fileName;
  }

  void save(List<Student> xs) throws IOException {
    try (BufferedWriter w = new BufferedWriter(new FileWr
      for (Student p : xs) {
        w.write(p.represent());
        w.newLine();
      }
    }
  }

  List<Student> load() throws IOException {
    try (BufferedReader r = new BufferedReader(new FileRe
      return r.lines() Stream<String>
        .map(Student::fromString) Stream<Student>
        .toList();
    }
  }
```

```java
package lesson17;

import java.io.*;
import java.util.List;

public class Intro1 {

  public static void loadAll(PizzaOps po) throws IOException, ClassNotFoundException {
    po.load()
      .forEach(System.out::println);
  }

public static void main(String[] args) throws IOException, ClassNotFoundException {
    PizzaOps po = new PizzaTxtFileOps( fileName: "pizza.txt");
    StudentBinFileOps so = new StudentBinFileOps( fileName: "student.bin");

    Pizza p1 = new Pizza( name: "M1",  size: 30);
    Pizza p2 = new Pizza( name: "M2",  size: 35);

    Student s1 = new Student( name: "Jim",  age: 33);
    Student s2 = new Student( name: "Jack",  age: 33);

    po.save(List.of(p1,p2));
    so.save(List.of(s1,s2));

    ///
```

```java
interface PizzaOps {

  void save(List<Pizza> xs) throws
  List<Pizza> load() throws IOExcep

}
```

A=Pizza 1

```
interface PizzaOps extends Ops<Pizza> {

  void save(List<Pizza> xs) throws IOException;
  List<Pizza> load() throws IOException, ClassNotFoundException;

}
```

A = Student 1

```
interface StudentOps extends Ops<Student>

  void save(List<Student> xs) throws IOException;
  List<Student> load() throws IOException, ClassNotFoundException;

}
```

```
interface Ops<A> {

  void save(List<A> xs) throws IOException;
  List<A> load() throws IOException, ClassNotFoundException;

}
```

```
interface DAO<A> {

  void save(List<A> xs) ;
  List<A> load() ;

}
```

~ CRUD

DAO ≈ DB

(A, A, A) → file

file → (A, A, ... A)

long String 8-10-12 UUID

id

```
public class Student
  String name;
  int age;
```

Create N w

Create 1 w

Read 1

Update 1

Delete 1

Read N w

CRUD

Student

mem

Student

db

```
interface DAO<A extends HasId> {

  default void save(A a) throws IOException {
    throw new IllegalArgumentException("TODO");
  }
  default void update(A a) throws IOException {
    throw new IllegalArgumentException("TODO");
  }
  default void delete(A a) throws IOException {
    delete(a.id());
  }
  default void delete(Long id) throws IOException {
    throw new IllegalArgumentException("TODO");
  }
```

```
public interface HasId {
  long id();
}
```

```
public static void loadAll(DAO<Pizza> po)
  po.loadAll()
    .forEach(System.out::println);
}
```

```
public class Pizza implements Serializable, HasId {
  long id;
  String name;
  int size;

  public Pizza(long id, String name, int size) {
    this.id = id;
    this.name = name;
    this.size = size;
  }

  @Override
  public long id() {
    return id;
  }
```