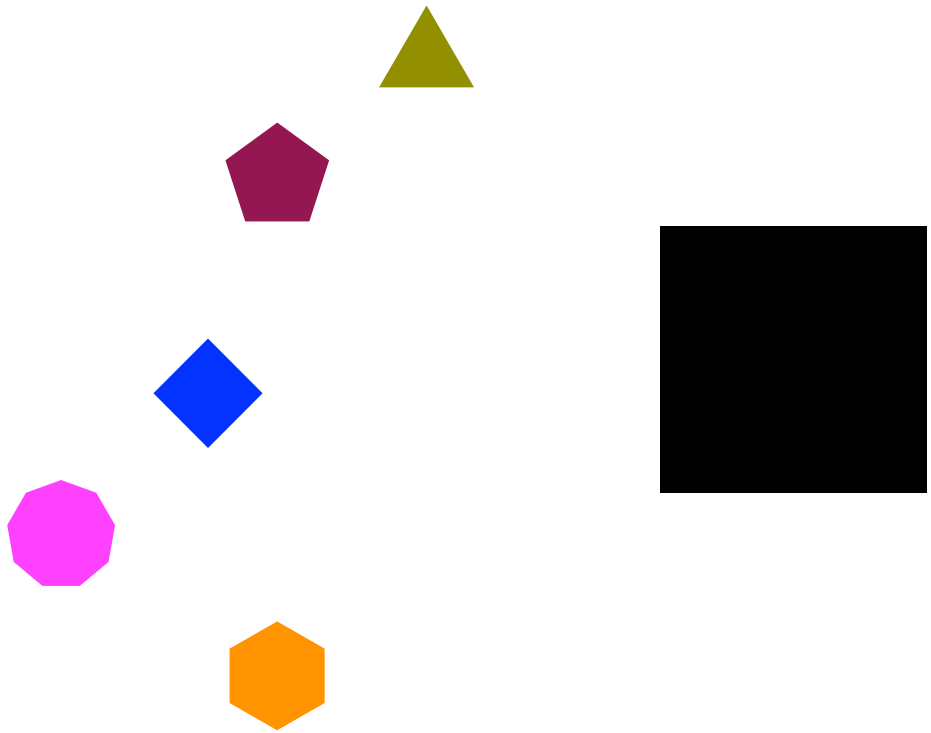



Welcome to
functional
programming.

programming =

Thinking

Coding

functional programming =

functional Thinking

+

functional Coding

Get Java 8 at
http://jdk8.java.net/download.html

Java 8

This course was created with an
Early Access edition of Java 8.

Java 6

Functions as Values

STORE functions in variables

PASS functions in parameters

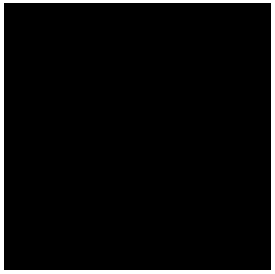
RETURN functions from other functions

The Calculation Engine

now in Java ~~8~~! 6!

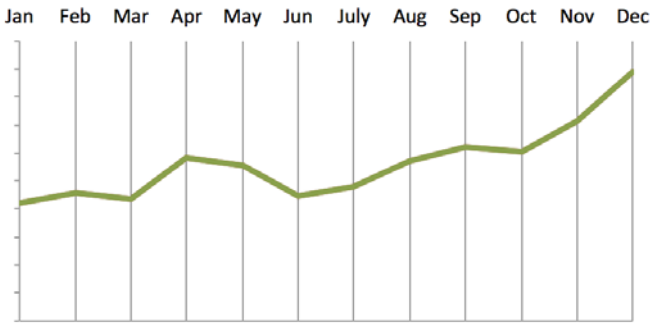
Sales

Incremental Costs



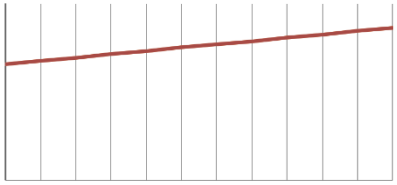
Profit!

Fixed Costs

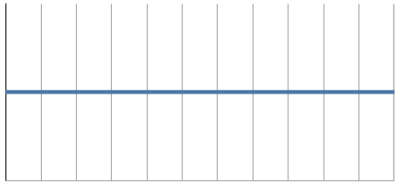


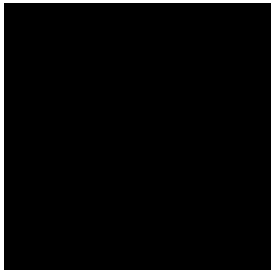
Sales

Incremental Costs



Fixed Costs





Profit!


lambda

Functions:

Meaning

evaluation has no outside effect

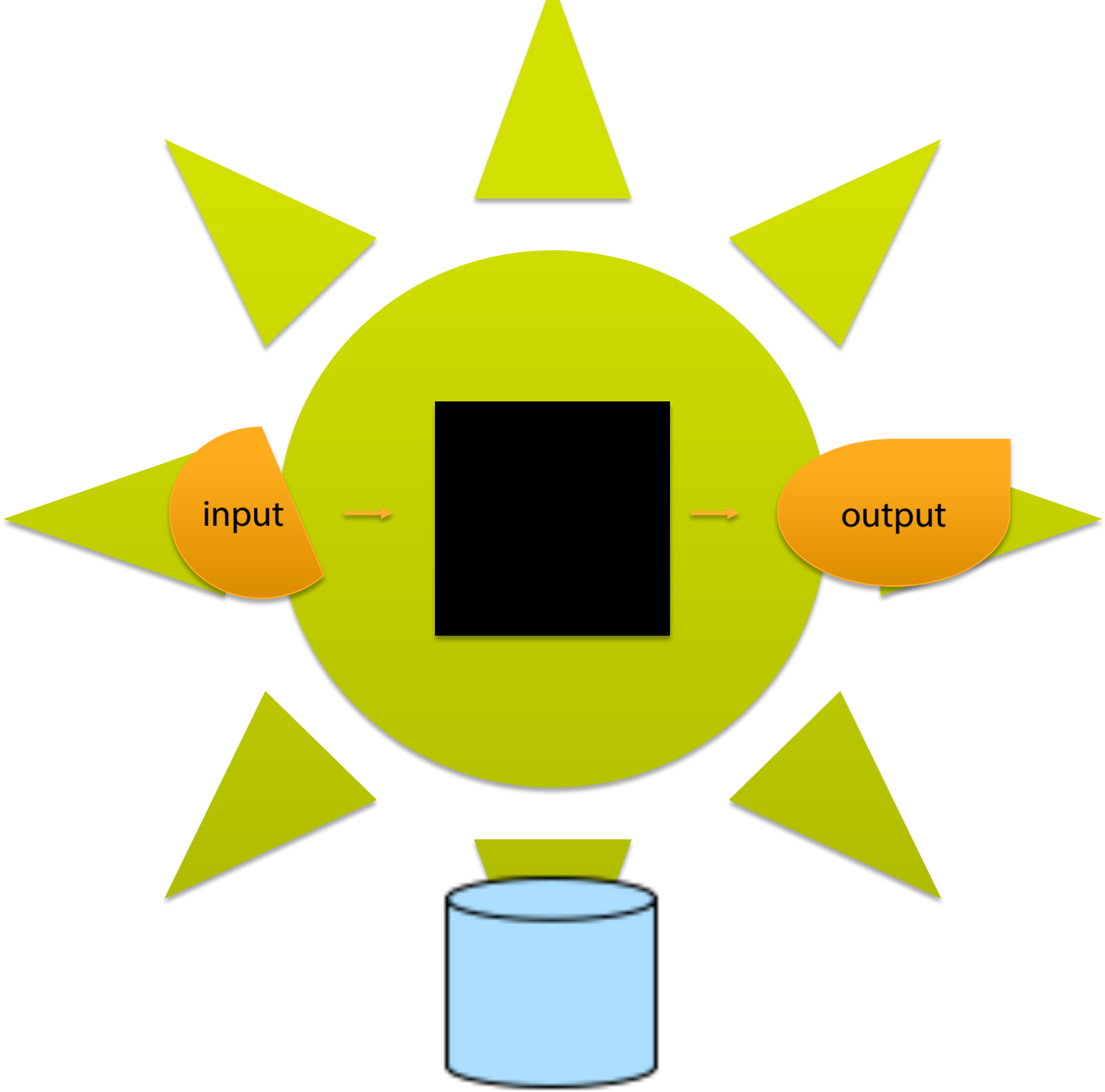
Mechanism

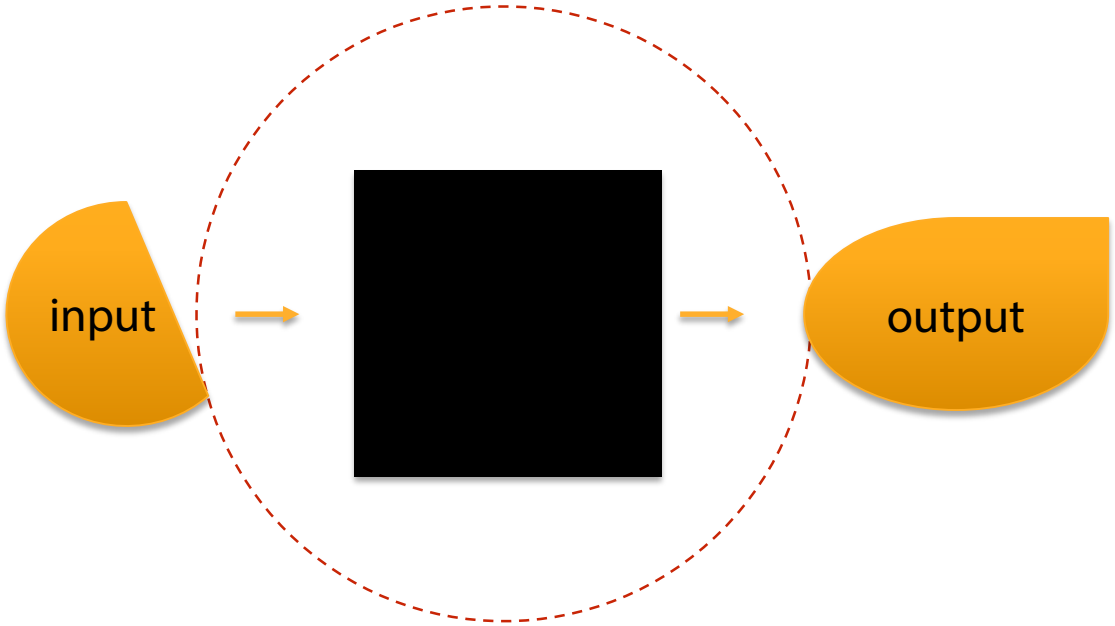
independent of any object instance

access global state

modify input

change the world


input
output



testable!

predictable!

access global state

modify input

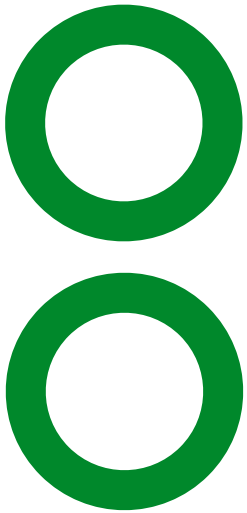
change the world

Data In, Data Out

surprise

s

Data In, Data Out



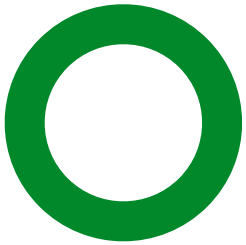
static methods

@FunctionalInterface

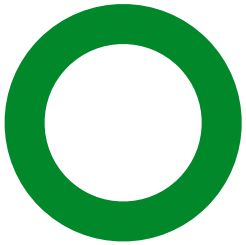
- type safety
- propertie s
- duplicatio n
- hierarch y
- simple r
- easy to add

duplicatio

n



clear naming



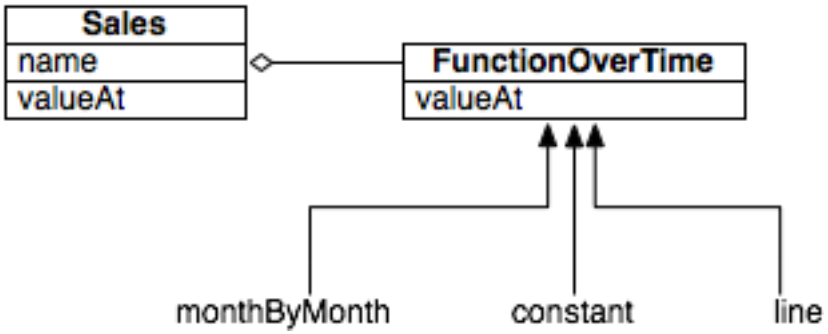
type safety

duplicatio n

deep inheritance

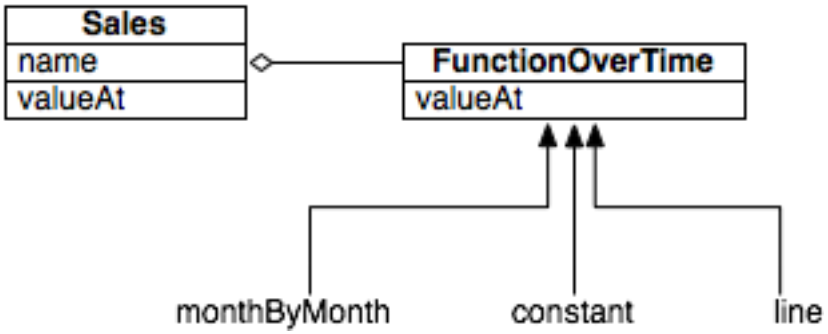
Strateg

y


Sales
name
valueAt
FunctionOverTime
valueAt
monthByMonth
constant
line

Strateg

y

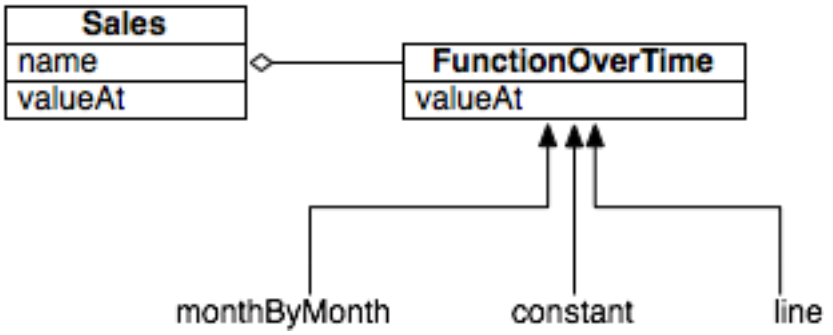



```
public interface FunctionOverTime {
  public Double valueAt(final Integer time);
}
```

```
 new FunctionOverTime() {
   @Override
   public Double valueAt(final Integer time) {
     return array[time - 1];
   }
 }
```

Strateg

y



Function<Integer, Double>

(time) -> array[time - 1];


You've learned
a lot today.

Functions as Values

STORE functions in variables

PASS functions in parameters

RETURN functions from other functions

Functions as Values

(also known as)

First-class Functions

Higher-order Programming

Lambda Expression

```
(time) -> array[time - 1];
```

Lambda

Lambdatron

Lambda Expression

```
(time) -> array[time - 1];
```

can see nearby variables

cannot change them!

can change nearby variables

Data In, Data Out

access global state

modify input

change the world

Data In, Data Out
(also known as)

Referential Transparency

```
FunctionOverTime c = (time) ->     ;          15.0

    "The result is " +                 c(100)
```

Data In, Data Out
(also known as)

Referential Transparency

"The result is " + 15.0

Data In, Data Out

(also known as)

Referential Transparency

Pure Functions

side

effects

Data In, Data Out

Functions

Data In, Data Out

Programs

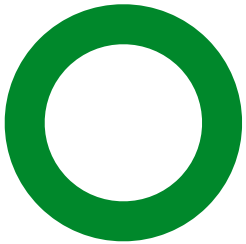
Functional Programming:

Meaning

think about evaluation before execution

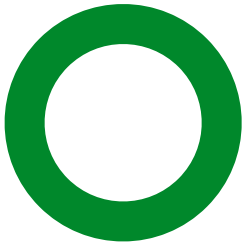
Mechanism

treat functions as values



readabl

e



reliabl

e

Java 8!