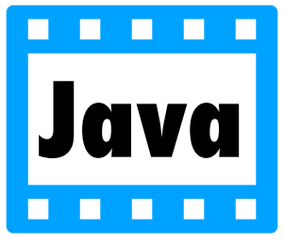


Accessors vs Mutators

presentation by Dr. K

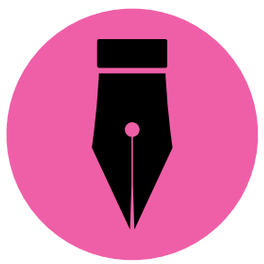
Accessor vs Mutator methods

My purpose is to **obtain information** about an object

I **do not** make any changes to the object

Accessor

My purpose is to **make changes** to an object



Mutator

The various meanings of "accessor method"

1. Methods used to obtain information about an object are known as **accessor methods**. One accessor method that you can use with strings is the length() method.

– Strings, The Java Tutorial

2. Constructs a PropertyDescriptor for a property that follows the standard Java convention by having getFoo and setFoo **accessor methods**.

– Property Descriptor, Java API

3. Often a setter is accompanied by a getter (also known as an **accessor**), which returns the value of the private member variable.

– Mutator Method, Wikipedia

The various meanings of "accessor method"

We use this one

1

> Methods used to obtain information about an object are known as **accessor methods**. One accessor method that you can use with strings is the length() method.

– Strings, The Java Tutorial

2

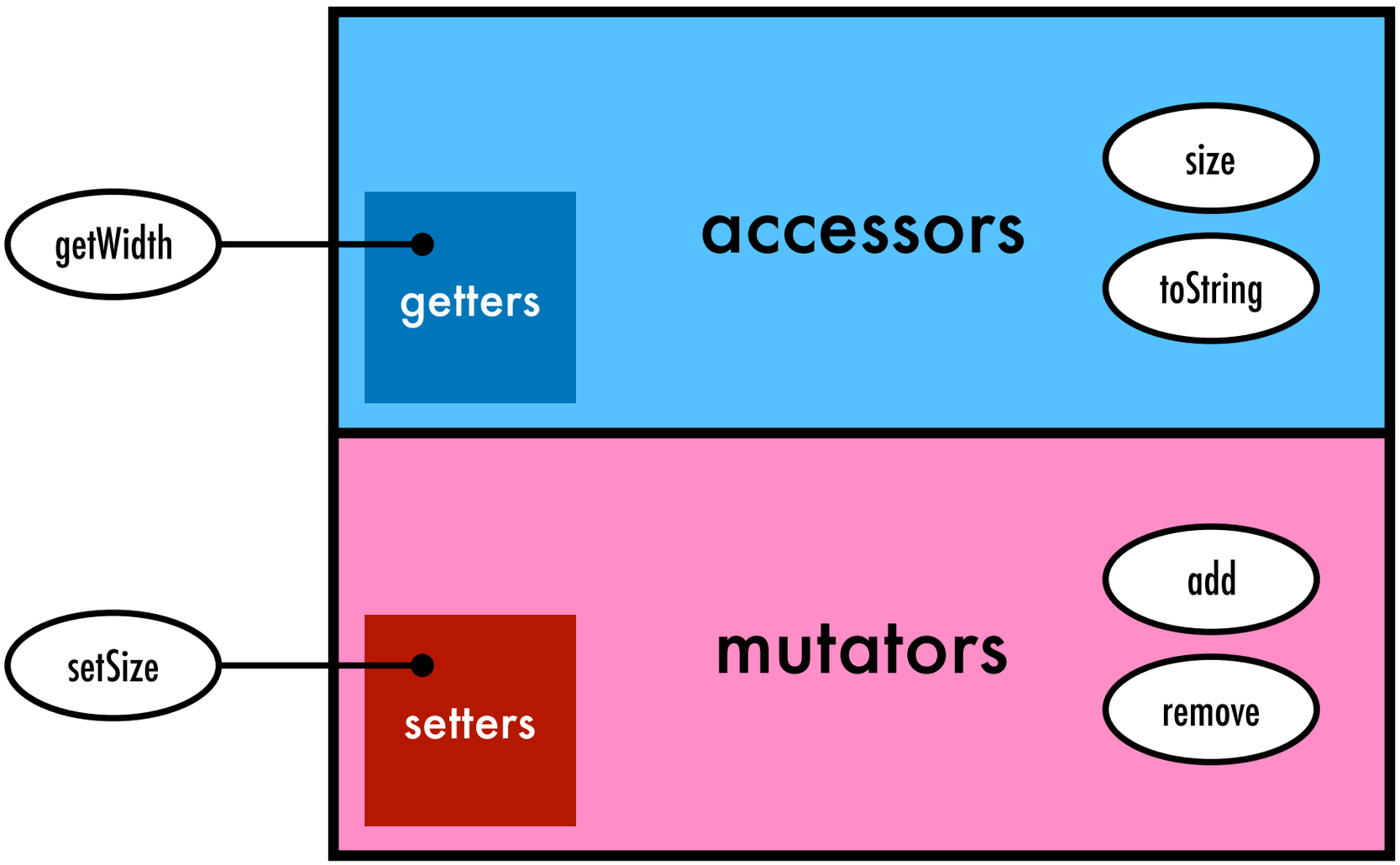
> ~~Constructs a PropertyDescriptor for a property that follows the standard Java convention by having getFoo and setFoo **accessor methods**.~~

– Property Descriptor, Java API

3

> ~~Often a setter is accompanied by a getter (also known as an **accessor**), which returns the value of the private member variable.~~

– Mutator Method, Wikipedia


getWidth
getters
accessors
size
toString
setSize
setters
mutators
add
remove

Accessors or Mutators

Type	Method and description
`int`	`depth()` Returns the number of elements in this stack.
`boolean`	`isEmpty()` Tests if this stack is empty.
`E`	`peek()` Returns a handle to the element at the top of this stack without removing it.
`E`	`pop()` Removes and returns the element at the top of this stack.
`void`	`push(E element)` Adds the specified element to the top of this stack.

Pause and Think

Which of the following methods are **accessors** and which are **mutators**? Why?

Type	Method and description
`int`	`depth()` Returns the number of elements in this stack.
`boolean`	`isEmpty()` Tests if this stack is empty.
`E`	`peek()` Returns a handle to the element at the top of this stack without removing it.
`E`	`pop()` Removes and returns the element at the top of this stack.
`void`	`push(E element)` Adds the specified element to the top of this stack.

Pause and Think

Which of the following methods are **accessors** and which are **mutators**? Why?

Type	Method and description	
`int`	**`depth`**`()` Returns the number of elements in this stack.	Accessor
`boolean`	**`isEmpty`**`()` Tests if this stack is empty.	Accessor
`E`	**`peek`**`()` Returns a handle to the element at the top of this stack without	Accessor
`E`	**`pop`**`()` Removes and returns the element at the top of this stack.	Mutator
`void`	**`push`**`(`**`E`**` element)` Adds the specified element to the top of this stack.	Mutator

Accessors or Mutators

Type	Method and description
`char`	**`charAt`**`(int index)` Returns the character value at the specified index.
`String`	**`concat`**`(`**`String`**` str)` Concatenates the specified string to the end of this string.
`boolean`	**`contains`**`(`**`String`**` str)` Returns true if and only if this string contains the specified string.
`char[]`	**`toCharArray`**`()` Converts this string to a new character array.
`String`	**`toUpperCase`**`()` Returns a copy of this string in which all characters are upper case.

Pause and Think

Which of the following methods are **accessors** and which are **mutators**? Why?

Type	Method and description
`char`	`charAt(int index)` Returns the character value at the specified index.
`String`	`concat(String str)` Concatenates the specified string to the end of this string.
`boolean`	`contains(String str)` Returns true if and only if this string contains the specified string.
`char[]`	`toCharArray()` Converts this string to a new character array.
`String`	`toUpperCase()` Returns a copy of this string in which all characters are upper case.

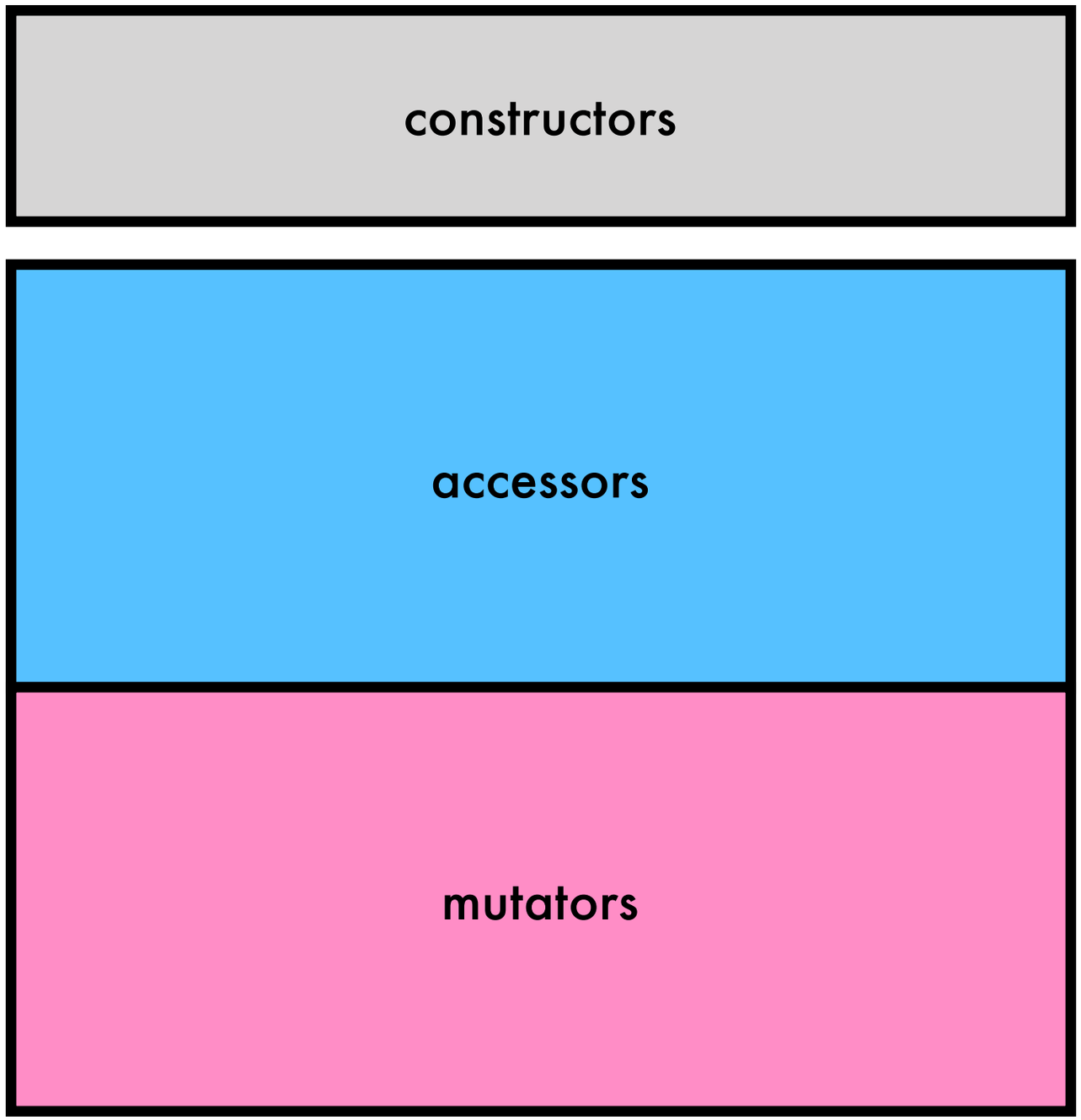
Pause and Think

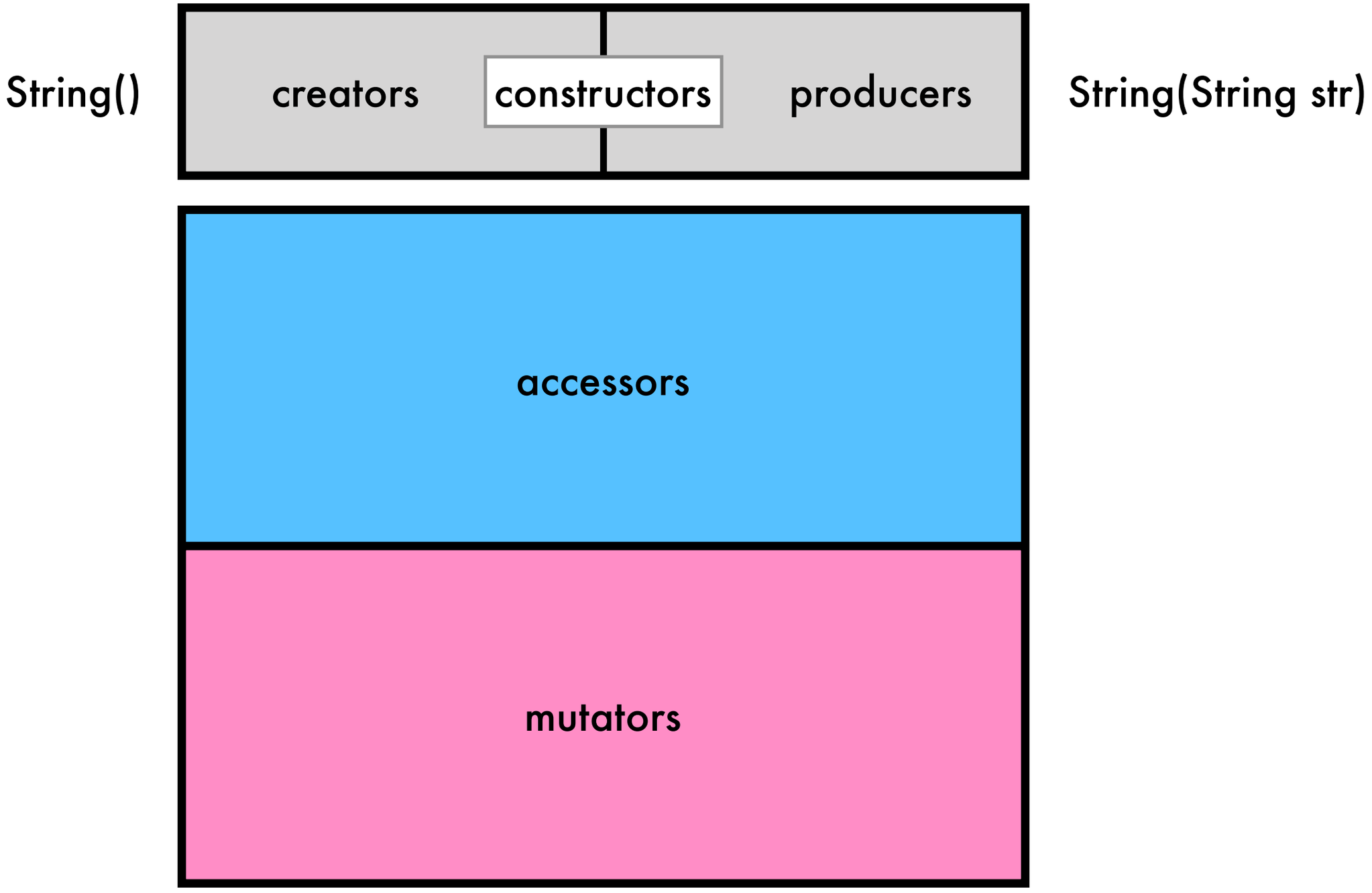
Which of the following methods are **accessors** and which are **mutators**? Why?

Type	Method and description	
`char`	**`charAt`**`(int index)` Returns the character value at the specified index.	Accessor
`String`	**`concat`**`(`**`String`**` str)` Concatenates the specified string to the end of this string.	Accessor
`boolean`	**`contains`**`(`**`String`**` str)` Returns true if and only if this string contains the specified strin	Accessor
`char[]`	**`toCharArray`**`()` Converts this string to a new character array.	Accessor
`String`	**`toUpperCase`**`()` Returns a copy of this string in which all characters are upper	Accessor

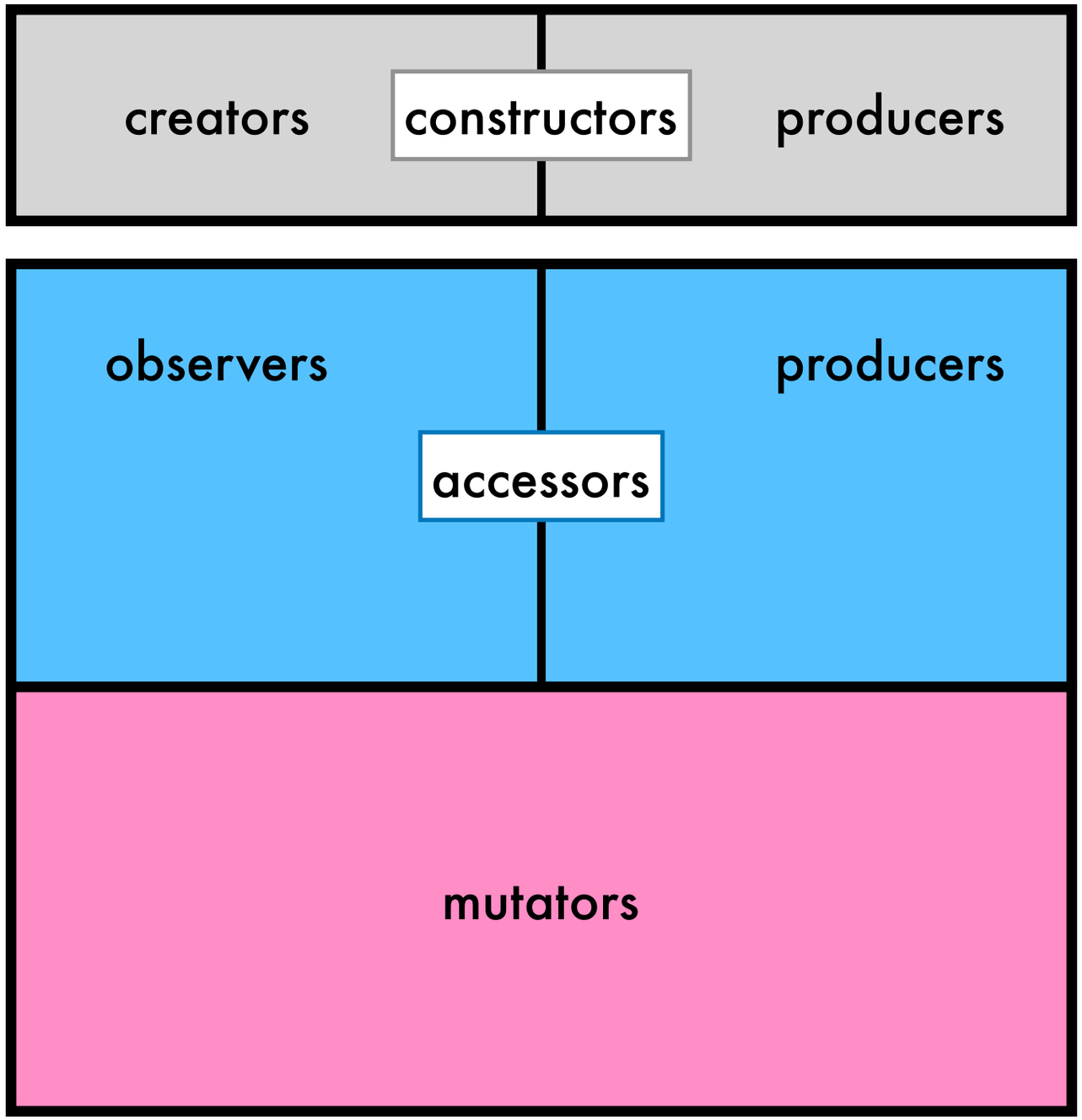
Liskov Categories

Creators	create objects of their type from scratch
Producers	create objects of their type based on other objects of their type
Mutators	modify objects of their type
Observers	return a type that is different from their type


constructors
accessors
mutators


String()
creators
constructors
producers
String(String str)
accessors
mutators

int indexOf(char)



String trim()

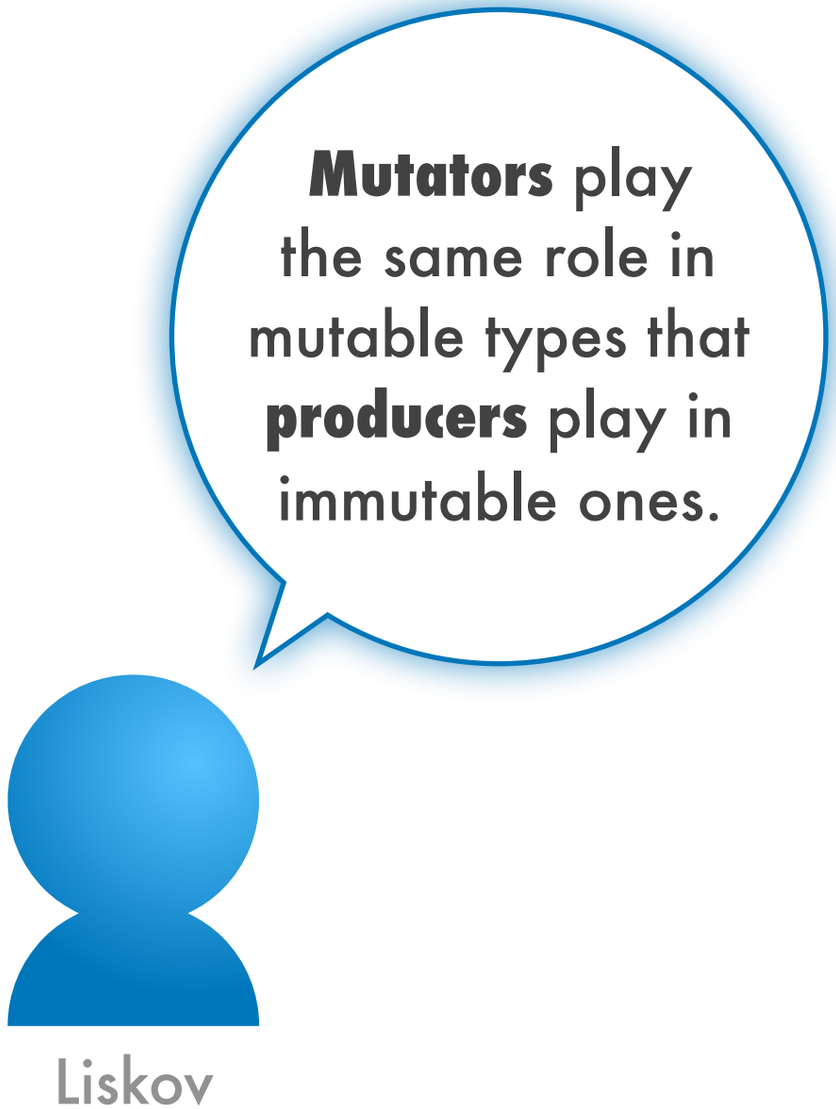
Producers or Observers

Type	Method and description
`char`	**`charAt`**`(int index)` Returns the character value at the specified index.
`String`	**`concat`**`(`**`String`**` str)` Concatenates the specified string to the end of this string.
`boolean`	**`contains`**`(`**`String`**` str)` Returns true if and only if this string contains the specified string.
`char[]`	**`toCharArray`**`()` Converts this string to a new character array.
`String`	**`toUpperCase`**`()` Returns a copy of this string in which all characters are upper case.

Producers or Observers

Type	Method and description	
`char`	**`charAt`**`(int index)` Returns the character value at the specified index.	Observer
`String`	**`concat`**`(`**`String`**` str)` Concatenates the specified string to the end of this string.	Producer
`boolean`	**`contains`**`(`**`String`**` str)` Returns true if and only if this string contains the specified strin	Observer
`char[]`	**`toCharArray`**`()` Converts this string to a new character array.	Observer
`String`	**`toUpperCase`**`()` Returns a copy of this string in which all characters are upper	Producer

Mutators and Producers



Mutators and Producers

```
/**
 * Adds the specified element
 * to the top of this stack.
 */
public void push(E element);
```

Pause and Think

How would you change this **mutator** method into a **non-mutator** method?

mutator

```
/**
 * Adds the specified element
 * to the top of this stack.
 */
public void push(E element);
```

Pause and Think

How would you change this **mutator** method into a **non-mutator** method?

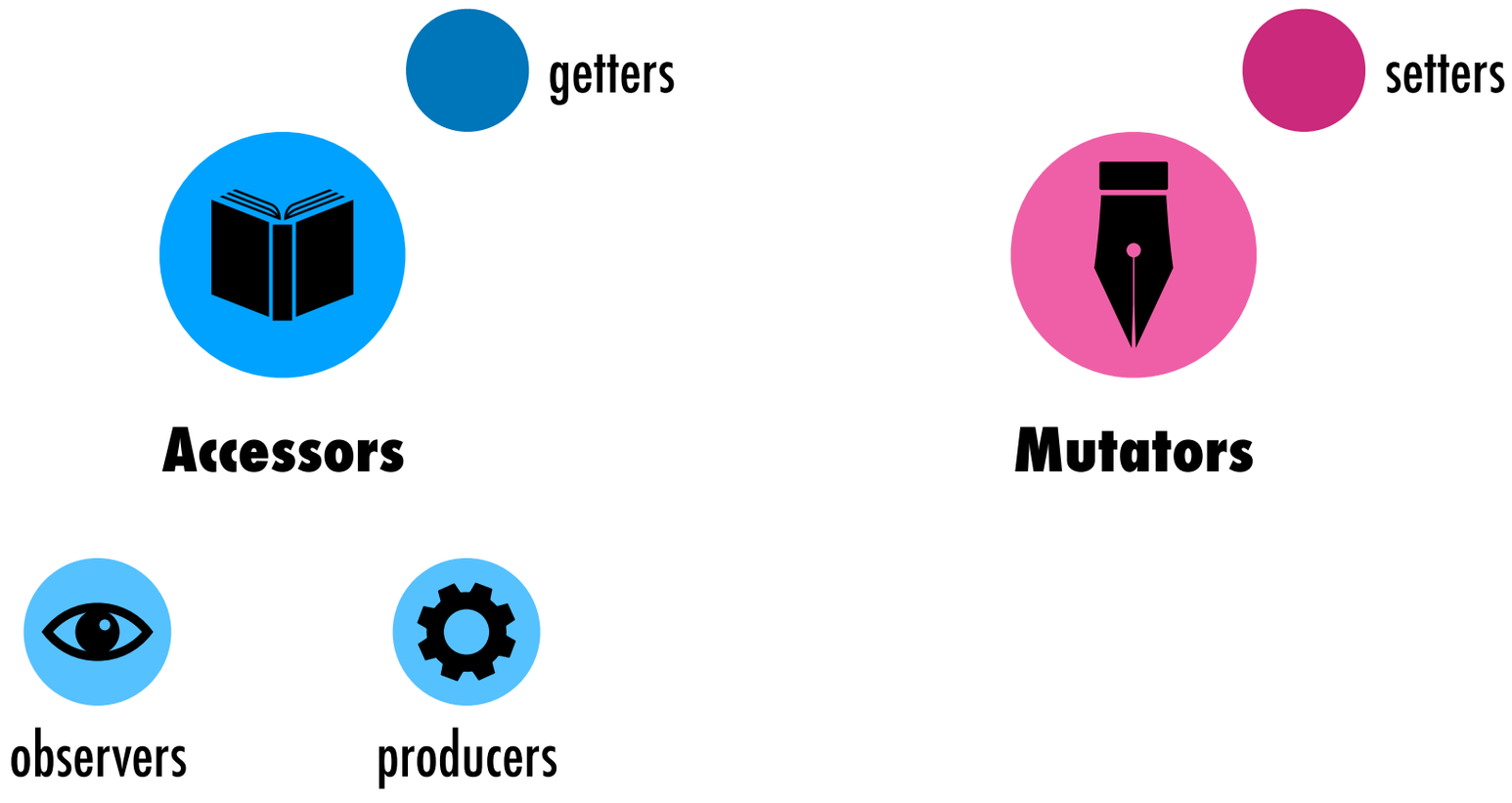
mutator

```
/**
 * Adds the specified element
 * to the top of this stack.
 */
public void push(E element);
```

producer

```
/**
 * Returns a copy of this stack with the
 * specified element added to the top.
 */
public Stack<E> push(E element);
```

Accessors vs Mutators



Why?

Create
Read
Update
Destroy

rwx

Why?

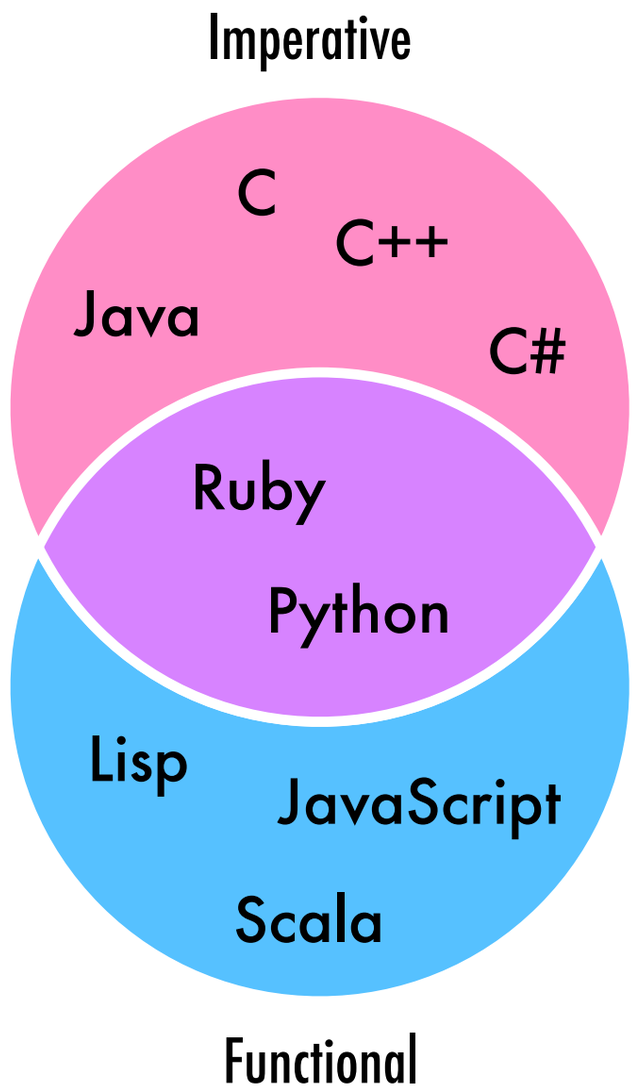
Create
Read
Update
Destroy

rwx

Accessors only:

- Conditions
- Assertions
- Immutable

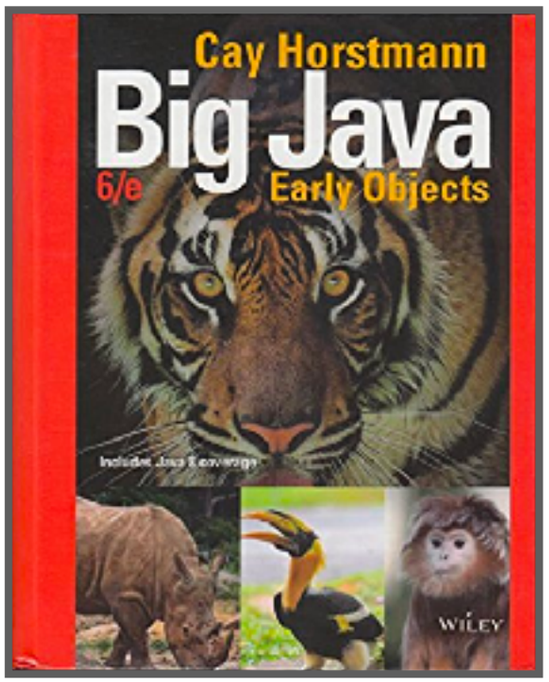
"Minimize Mutability"
— Joshua Bloch

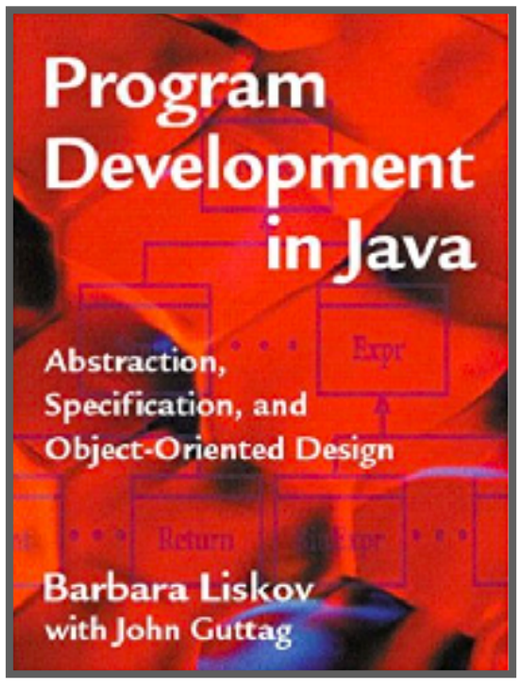


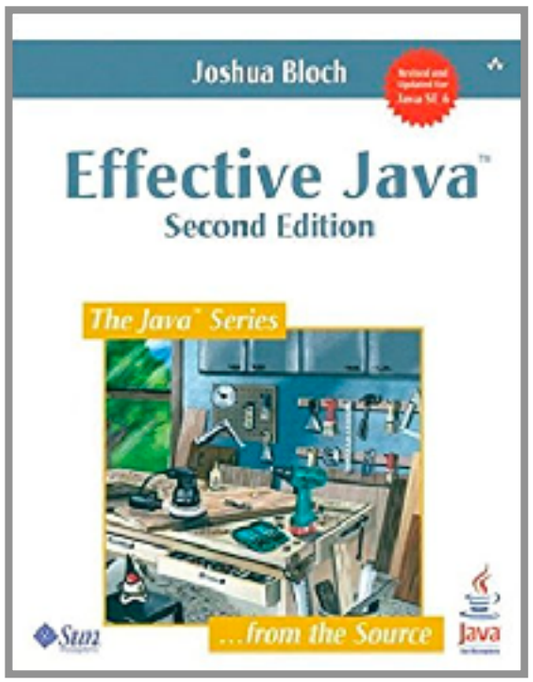
Take-Aways

- **Accessors** read information while **Mutators** modify objects
- Getters get properties and setters set them
- Some use *accessor* to refer to both getters and setters
- Richer categorizations account for other method types
- Immutable classes do not have mutator methods!

Look It Up


Cay Horstmann
Big Java
6/e
Early Objects
WILEY


Program Development in Java
Abstraction, Specification, and Object-Oriented Design
Barbara Liskov
with John Guttag


Joshua Bloch
Effective Java
Second Edition
The Java Series
...from the Source