



ORACLE

JAVA PUZZLE BALL

Lesson 4

Lambda Puzzles 1-5

Lambda Expressions, Streams, and Functional Programming



www.oracle.com/goto/JavaGame

Did You Try Lambda Puzzles 1-7?

- Can you identify use-cases for lambda expressions?

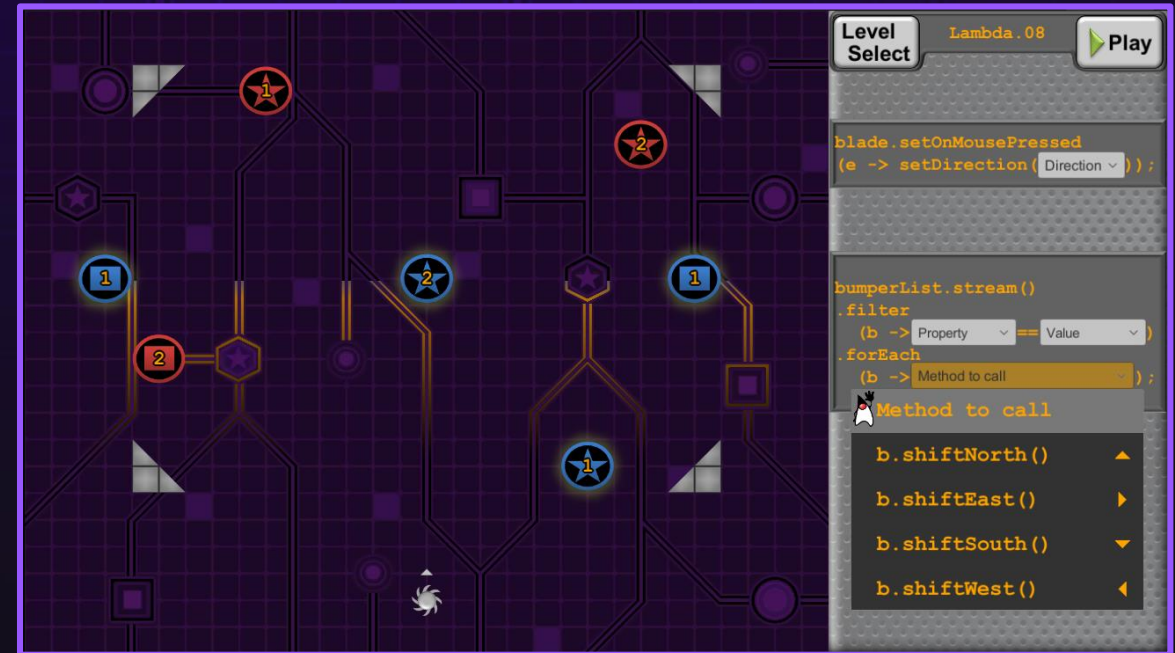
- How do these logic operators work?

==

!=

&&

||



These are Lambda Expressions

```
blade.setOnMousePressed  
(e -> setDirection( Dir.NE )) ;
```

```
bumperList.stream()  
.filter  
(b -> b.getShape() == Shape.STAR )  
.forEach  
(b -> b.setShape(Shape.RECT) ) ;
```

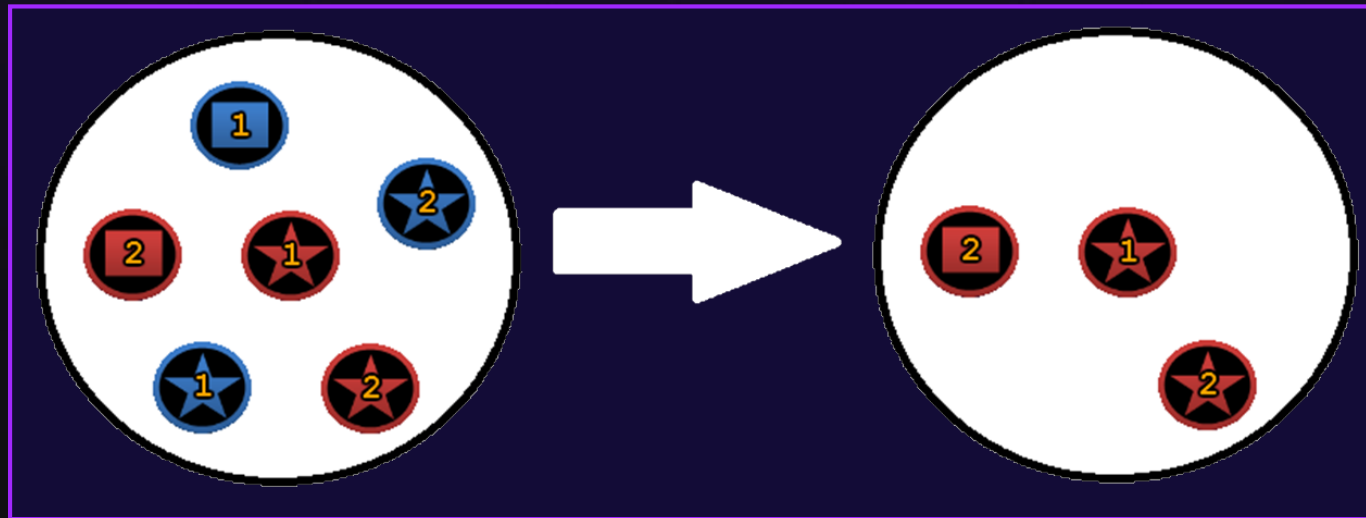
Lambda Use-Case 1

- Lambda expressions handle mouse, keyboard, and other input events.
- An event occurs when the ball/blade is pressed.
- Following the event, the lambda expression executes:
 - `e` represents the event.
 - `setDirection()` is a method which changes the ball's direction field.
 - `Dir.NE` becomes the new value of the direction field.

```
blade.setOnMousePressed  
(e -> setDirection( Dir.NE  ▾ ));
```


Lambda Use-Case 2

- Lambda expressions let you easily work with a collection of objects.
- As you play, you perform the same logic as the lambda code:
 - Take a collection of many bumpers.
 - Identify bumpers based on certain properties (color).
 - For each bumper matching that criteria, perform an action on it (Destroy/Preserve).



Examining the Java Code

- `bumperList` is the collection of bumpers.
- `b` represents any bumper object in the collection.
- A `filter` isolates objects based on their properties (shape, color, number).
- A method is called `forEach` object that matches the filter criteria.

```
bumperList.stream()  
  .filter  
    (b -> b.getShape() == Shape.STAR )  
  .forEach  
    (b -> b.setShape(Shape.RECT) );
```

Filter Statements

- `==` checks if a property **is equal** to a value.
- `!=` checks if a property **is not equal** to the value.
- Lambda Puzzle 7 shows multiple filters can be chained together.
 - Objects must pass both filters

```
bumperList.stream()
    .filter
        (b -> b.getColor() == Color.BLUE )
    .filter
        (b -> b.getShape() != Shape.STAR )
    .forEach
        (b -> b.shiftEast() ) ;
```

The diagram illustrates the structure of filter statements in a Java Stream. It shows a sequence of filter operations. The first filter is `(b -> b.getColor() == Color.BLUE)`, and the second is `(b -> b.getShape() != Shape.STAR)`. Handwritten labels 'Property' and 'Value' are placed above the first filter. A vertical line from 'Property' points to the lambda expression `b -> b.getColor()`. Another vertical line from 'Value' points to the comparison value `Color.BLUE`. The code is enclosed in a purple border.

Compound Logic

- Sometimes programs need to compare several values at once.
- Lambda Puzzle 6 shows compound logic applied in filters.
- With `&&`, both criteria must be true for an object to pass through the filter.
- With `||`, either criteria can be true for an object to pass through the filter.

```
bumperList.stream()  
  .filter(b ->  
    b.getNum() == 1  
    || b.getColor() == Color.RED  
  )  
  .forEach  
    (b -> b.shiftSouth());
```


Summary of Logic Operators

==

Are two values
equal?

!=

Are two values
not equal?

&&

And And

||

Or Or

What is Functional Programming?

- Remember, variables in Java store more than just numbers.
- Functional programming is about storing and passing around functionality and logic.
 - Save a lambda expression as a variable.
 - Reference that variable later.
 - Pass the variable or lambda logic between different bits of code.

```
public class ExampleClass {  
    //Fields  
    private Consumer<ImageView> lambdaExample = (e -> setDirection(Dir.NE));  
    private int x = 0;  
    private int y = 1;  
    private int z = 2;  
  
    ...  
}
```

Further Learning

Edit code and learn more in the multi-day version of this course:

- www.oracle.com/goto/JavaGame

