Lambda Expressions

Department of Computer Science
University of Maryland, College Park
Lambda Expressions

• Lambda expression - can be seen as a concise approach to define an anonymous class instance

• **Functional interface** - Java interface with a **single** abstract method (default methods are fine)

• Example:

  ```java
  public interface Task {
    public int compute(int x);
    public default int version() {
      return 10;
    }
  }
  ```

• Java provides support for lambda expressions **only with functional interfaces**

• Compiler treats a lambda expression as an object created from an anonymous class
Lambda Expressions

Example:

```java
public interface Task {
    public int compute(int x);

    public default int version() {
        return 10;
    }
}

/* Using anonymous class instance */
Task anonymousClassInstance = new Task() {
    public int compute(int x) {
        return x + x;
    }
};
System.out.println(anonymousClassInstance.compute(10));

/* Using lambda expression */
Task lambda = x -> x + x;
System.out.println(lambda.compute(10));
```
Lambda Expressions

• Lambda Expression Syntax

(type1 parameter1, type2 parameter2, …) -> expression

OR

(type1 parameter1, type2 parameter2, …) -> { statements }

• The parameter type can be inferred by the compiler
• Parenthesis can be dropped if there is only one parameter
• Lambda expressions cannot be defined for abstract classes
• Example: LambdaBasics.java
• https://docs.oracle.com/javase/tutorial/java/javaOO/lambdaexpressions.html