Java Lambda Expression Practice Worksheet with Examples & Explanations

Day 1: Basics of Lambda

Task 1: Runnable using Lambda
Instead of using an anonymous inner class, use a lambda expression to create a Runnable.

Example:
```
Runnable r = () -> System.out.println("Running with Lambda");
r.run();
```
Explanation: This lambda implements the run() method from the Runnable interface.

Task 2: Functional Interface
Create your own functional interface and use a lambda to implement it.

Example:
```
@FunctionalInterface
interface Sayable {
    void say(String msg);
}
Sayable s = message -> System.out.println("Message: " + message);
s.say("Hello Lambda!");
```
Explanation: Lambda is used to implement the single abstract method say().

Day 2: Lambda with Collections

Task 1: Iterating List with Lambda
Use lambda to iterate over a list using forEach().

Example:
```
List names = Arrays.asList("Sonali", "Snehal", "Pooja");
names.forEach(name -> System.out.println("Hi " + name));
```
Explanation: Lambda is passed as a Consumer to forEach().

Task 2: Sorting List
Sort a list using lambda inside Collections.sort or List.sort.

Example:
```
List items = Arrays.asList("Banana", "Apple", "Mango");
items.sort((a, b) -> a.compareTo(b));
System.out.println(items);
```
Explanation: Lambda provides Comparator logic.

Day 3: Lambda with Streams

Task 1: Filter Even Numbers
Use Stream API and lambda to filter even numbers.

Example:

```
List nums = Arrays.asList(1, 2, 3, 4, 5);
nums.stream().filter(n -> n % 2 == 0).forEach(System.out::println);
```
Explanation: Lambda is used as Predicate in filter().

Task 2: Map to Uppercase
Convert strings in a list to uppercase using map().

Example:
```
List words = Arrays.asList("java", "lambda");
List upper = words.stream().map(String::toUpperCase).toList();
System.out.println(upper);
```
Explanation: Lambda is replaced with method reference String::toUpperCase.

Day 4: Custom Functional Interfaces

Task: Arithmetic Operations
Define a custom interface and implement add, multiply using lambda.

Example:
```
@FunctionalInterface
interface Operation {
    int apply(int a, int b);
}
Operation add = (a, b) -> a + b;
Operation mul = (a, b) -> a * b;
System.out.println("Sum: " + add.apply(3, 5));
System.out.println("Mul: " + mul.apply(3, 5));
```
Explanation: Lambda implements apply() with different logic.

Day 5: Built-in Functional Interfaces

Task 1: Predicate
```
Predicate isShort = str -> str.length() < 5;
System.out.println(isShort.test("Java"));
```
Explanation: Used to test a condition.

Task 2: Function
```
Function toLength = s -> s.length();
System.out.println(toLength.apply("Lambda"));
```
Explanation: Takes input and returns a result.

Task 3: Consumer
```
Consumer printer = s -> System.out.println("Val: " + s);
printer.accept("Hello");
```
Explanation: Performs action but returns nothing.

Task 4: Supplier
```
Supplier random = () -> Math.random();
System.out.println(random.get());
```
Explanation: Supplies a value with no input.