[실습문제 1] 람다식 구현1

문제

LamdaPractice1코드를 확인 후 , printStrings내부에 다음 조건을 만족하는 람다식들을 구현하시오.

1. **길이 3 이하인 문자열만 출력**되는 람다식 구현.
2. 문자열에 `"a"`가 **포함된 단어만 출력**하는 람다식 구현
3. 문자열이 `"w"`로 **시작하는 것만 출력**하는 람다식 구현
4. 문자열 길이가 **짝수면서 문자열에 `"a"`가 포함된 단어만 출력**하는 람다식 구현

```
public class LamdaPractice1 {
	public static void main(String[] args) {
		String[] arr = { "java", "lambda", "hi", "functional", "wow" };
		printStrings(이부분 구현);

	}
	public static void printStrings(String[] arr, StringChecker checker) {
		for (String str : arr) {
			if (checker.check(str)) {
				System.out.println(str);
			}
		}
	}

	@FunctionalInterface
	interface StringChecker {
		boolean check(String str);
	}

}
```

[실습문제 2] 람다식 구현2

문제

LamdaPractice2코드를 확인 후 , 다음 조건을 만족하는 람다식들을 구현하시오.

1. `Calculator<Integer>`로 **두 수의 곱셈 결과**를 반환하는 람다식을 작성하시오.
2. `Calculator<Integer>`로 **두 수 중 큰 값 반환(max)** 람다식을 작성하시오.
3. `Calculator<String>`로 `"hello"`, `"world"` 입력 시 `"hello-world"`가 나오도록 작성하시오.

```
public class LamdaPractice2 {

	public static void main(String[] args) {

		Calculator<Integer> adder = (1번 구현)

		Calculator<Integer> findMax = (2번 구현)

		Calculator<String> combiner = (3번 구현)

	}


	@FunctionalInterface

	interface Calculator <V>{

		V operate(V v1, V v2);

	}

}
```

[실습문제 3] 람다식 구현3

문제

LamdaPractice3코드를 확인 후 , 다음 조건을 만족하는 람다식들을 구현하시오.

1. `MyFunction<String, String>`을 만들어 입력 문자열을 **대문자로 변환**하여 반환하시오.
2. `MyFunction<String, Boolean>`을 만들어 문자열 길이가 **5 이상인지 여부**를 반환하시오.
3. `MyFunction<String, String>`을 만들어 `"lambda"` 입력 시 `"l-a-m-b-d-a"` 형태로 반환하시오.

```
public class LamdaPractice3 {

	public static void main(String[] args) {

		MyFunction <String , String> first = (1번 구현)

		MyFunction <String, boolean> second = (2번 구현)

		MyFunction <String, String> third = (3번 구현)

	}


	@FunctionalInterface

	interface MyFunction<V , B>{

		B apply(V v);

	}

}
```

[실습문제 4] 람다식 구현4

문제

LamdaPractice4코드를 확인 후 , 다음 조건을 만족하는 람다식들을 구현하시오.

1. 학생의 이름과 점수를 출력하고, 만약 점수가 **90 이상이면 “A”, 80 이상이면 “B”, 70이상이면 “C” , 60점 이상이면 “D”** 학점을 함께 출력하시오.

```
public class LamdaPractice4 {

	public static void main(String[] args) {

		MyFunction2<String, Integer> printer = (1번 구현)


		printer.accept("홍길동", 95); // 홍길동님의 점수는 A학점(95)입니다.

		printer.accept("이순신", 88);  // 이순신님의 점수는 B학점(88)입니다

	}


	@FunctionalInterface

	interface MyFunction2 <K , A>{

		void accept(K k, A a);

	}

}
```

[실습문제 5] 람다식 구현5

문제

`getOperator(String type)` 메서드를 완성하여,
문자열로 전달된 연산자(`"+", "-", "*", "/", "%"`)에 따라
적절한 `BiFunction<Integer, Integer, Integer>` 람다식을 반환하도록 구현하시오.

```
public class LamdaPractice5 {
	public static void main(String[] args) {
		BiFunction<Integer, Integer, Integer> op = getOperator("+");
		System.out.println(op.apply(10, 5));  // 15

		op = getOperator("*");
		System.out.println(op.apply(10, 5));  // 50
	}

	public static BiFunction<Integer, Integer, Integer> getOperator(String type) {
	   //이부분 코드 구현
	}
}
```

[실습문제 6] 람다식 구현6

문제

`getComparator(String mode)` 메서드를 완성하여,
문자열 리스트를 **정렬 기준(mode)에 따라 정렬할 수 있도록** 구현하시오.

```
public class LamdaPractice6 {

	public static void main(String[] args) {
		List<String> names = Arrays.asList("Jin", "Suga", "RM", "J-Hope", "V");

		Comparator<String> comp = getComparator("length");
		names.sort(comp);
		System.out.println(names); // 길이순 정렬

		comp = getComparator("reverse");
		names.sort(comp);
		System.out.println(names); // 알파벳 역순 정렬
	}
	public static Comparator<String> getComparator(String mode) {
	   //코드 구현
	}
}
```