Systems for Design and Implementation

Outline

- Reactive programming
- Exam

Reactive programming

- Marble diagrams
- map, do , filter, take, first, skip, takeLast, last
- concat, startWith
- merge, combineLatest, zip
- switch, mergeAll, concatAll
- switchMap, mergeMap, concatMap
- Subject, BehaviorSubject, ReplaySubject, AsyncSubject

Exercises

What is the output of the following code sequences? **Explain** your choice.

1.

```
1. class Person {
2.     int age;
3. }
4. class PersonChecker {
5.     static void check(Person p, Predicate<Person> predicate) {
6.         System.out.println(predicate.test(p) ?
7.                            "drinks beer" : "doesn't drink beer");
8.     }
9. }
10.    //main
11.    Person p = new Person();
12.    p.age = 18;
13.    PersonChecker.check(p, p1 -> p1.age >= 18);
```

A. drinks beer
B. doesn't drink beer
C. Exception on line 13
D. Exception on line 6
E. The code does not compile

2.

```
Comparator<Integer> c = (o1, o2) -> o2 - o1;
List<Integer> list = Arrays.asList(5, 4, 7, 1);
Collections.sort(list, c);
System.out.println(Collections.binarySearch(list, 1));
```

A. 3
B. 4
C. 0
D. -1
E. The behavior is undefined

3.

```
1. Stream.generate(() -> "abracadabra")
2.       .filter(n -> n.length() == 4)
3.       .limit(2)
4.       .sorted()
5.       .forEach(System.out::println);
6. System.out.println("hello");
```

A. hello
B. By commenting out line 2, the output is “abracadabra” printed twice
C. By commenting out lines 2 and 3, the code prints “abracadabra” endlessly
D. The code prints “abracadabra” twice and then “hello”
E. Execution hangs

4.

```
@Entity
@Getter
@Setter
@ToString
public class Person implements
Serializable {
  @Id
  @GeneratedValue
  private Long id;
  private String name;
  @OneToOne
  private Address address;
}
@Entity
@Getter
@Setter
public class Address implements
Serializable {
  @Id
  @GeneratedValue
  private Long id;
  private String city;
}
public interface PersonRepository
extends JpaRepository<Person, Long> {
}
```

```
//main
//context is correctly initialised

PersonRepository personRepository =
context.getBean(
      PersonRepository.class);

personRepository.findAll()
            .forEach(System.out::println);

/*
There are 4 persons in the DB. The first 2
persons have the address in “cluj”, the last 2
persons have the address in “london”.

A. There will be 4 generated select
statements.
B. There will be 3 generated select
statements.
C. There will be 5 generated select
statements.
D. There will be 1 generated select
statement.
E. An exception is thrown.
*/
```