```java
package quiz2;

import java.util.function.Function;
import java.util.function.BiFunction;
import java.util.function.Consumer;
import java.util.function.BiConsumer;
import java.util.Comparator;

public class Examples {

    //object:: instanceMethod . Given an object ob and an instance method math()
    in ob
    //x -> ob.math( x)
    //ob ::math

    //Class:: staticMethod
    //ClassName and one of its static methods math(), the lambda expression
    //(x,y) -> ClassName.math(x,y)
    //ClassName::math

    //Class instanceMethod . Given a class ClassName and one of its instance
    methods math()
    //(x,y) -> x.math ( y)
    //ClassName::math
```

Simple Solution

```java
    //type: Class::instanceMethod
    Function<Employee, String> e1 = (Employee e) -> e.getName();
    Function<Employee, String> e2 = Employee::getName;
    Function<Employee, String> e3 = new Function<Employee, String>() {
        @Override
        public String apply(Employee t) {
            return t.getName();
        }
    };
```

①

```java
    //type:
    //(String s) -> s.toUpperCase()
```

②

```java
    //type:
    //(Employee e) -> e.setName("default");
```

③

```java
    //type:
    //(Employee e,String s) -> e.setName(s);
```

4

```
    //type:
    //(String s1,String s2) -> s1.compareTo(s2);
```

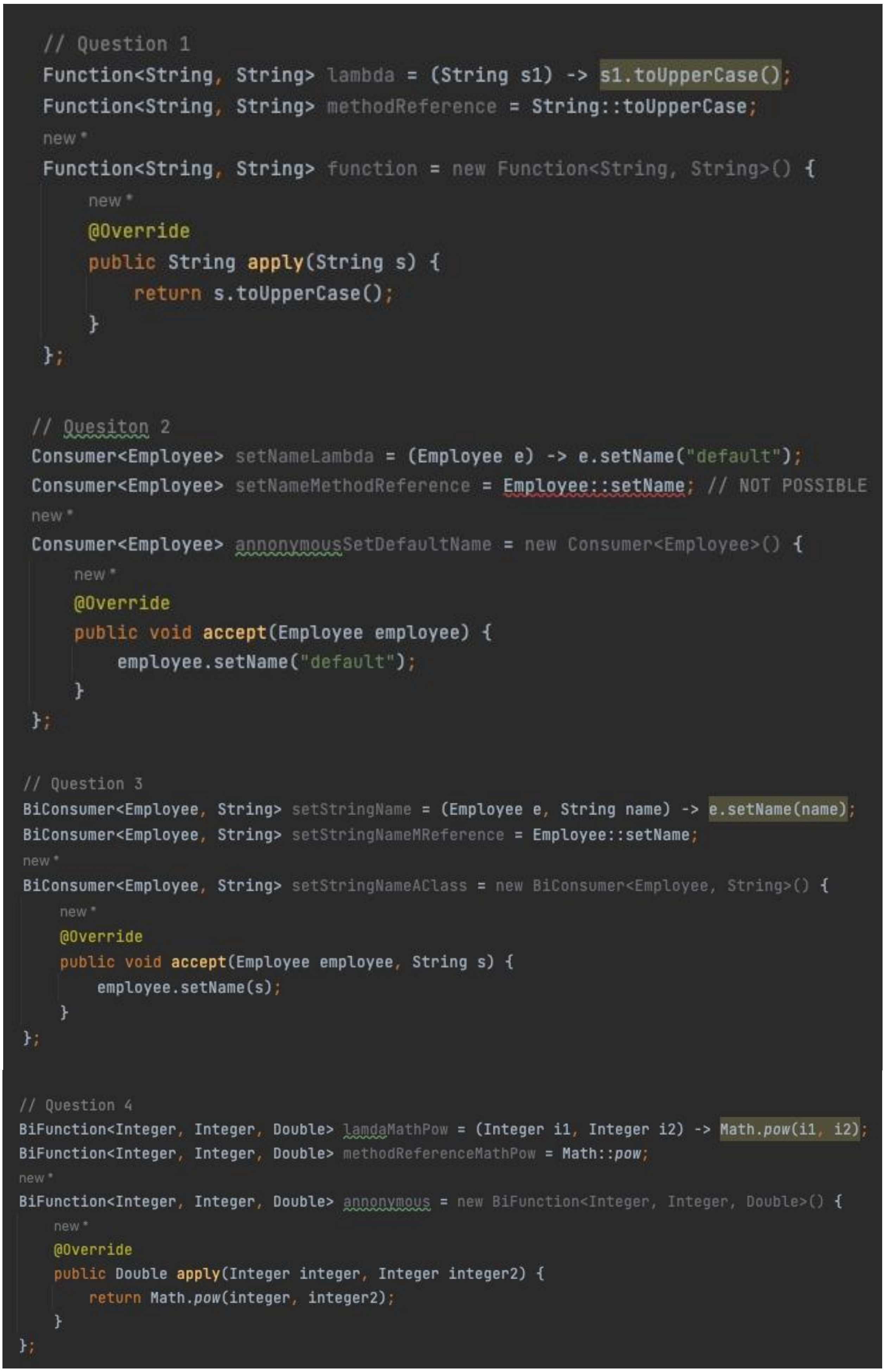
5

```
    //type:
    //(Integer i1,Integer i2) -> Math.pow(i1,i2);
```

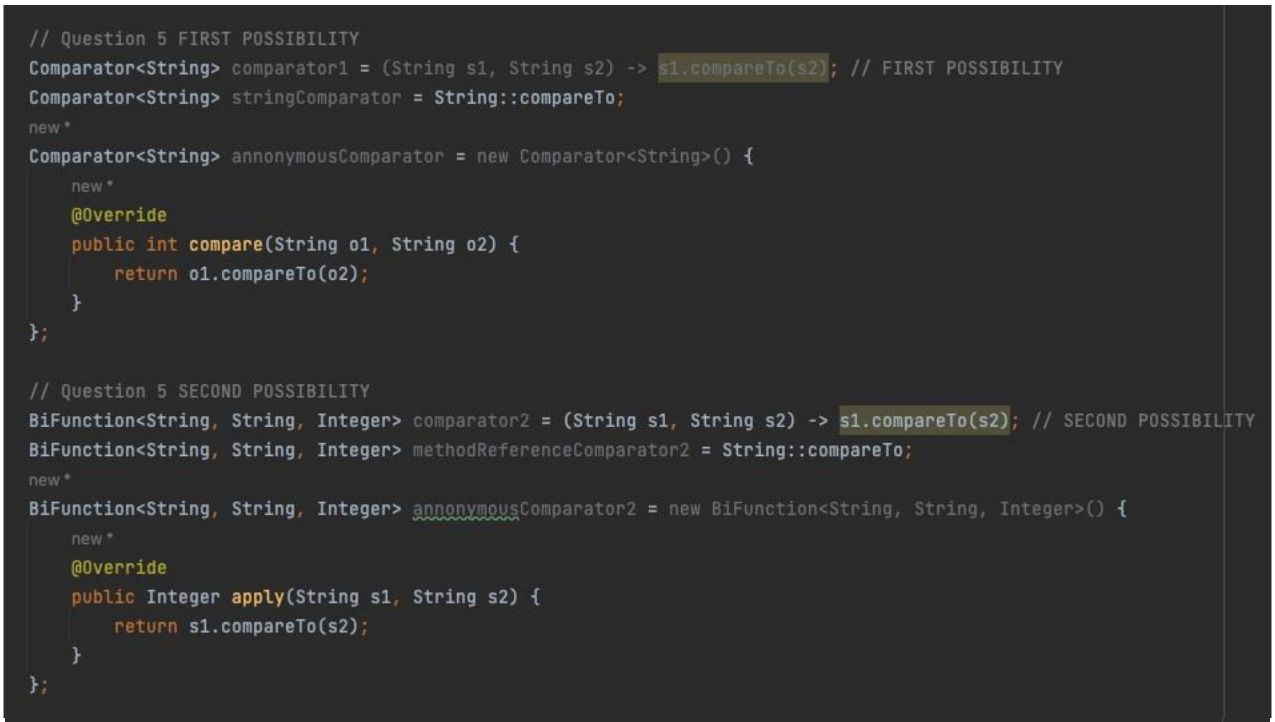
6

```
    //type:
    EmployeeNameComparator comp = new EmployeeNameComparator();
    //(Employee e1, Employee e2) -> comp.compare(e1,e2);
```

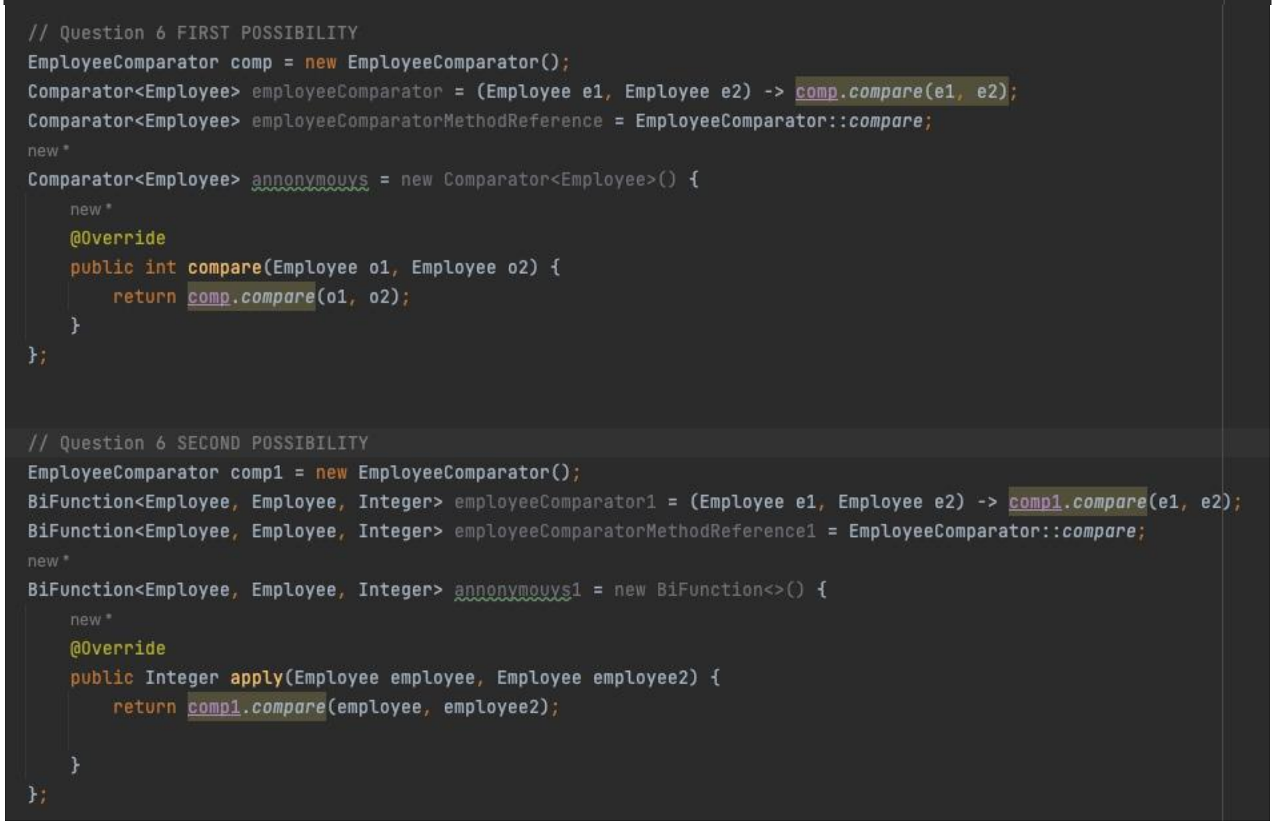
```
    public void evaluator() {
        System.out.println(e1.apply(new Employee("name",100)));
    }

    public static void main(String[] args) {
        Examples e = new Examples();
        e.evaluator();
    }

}
```



```java
// Question 1
Function<String, String> lambda = (String s1) -> s1.toUpperCase();
Function<String, String> methodReference = String::toUpperCase;
new *
Function<String, String> function = new Function<String, String>() {
    new *
    @Override
    public String apply(String s) {
        return s.toUpperCase();
    }
};

// Quesiton 2
Consumer<Employee> setNameLambda = (Employee e) -> e.setName("default");
Consumer<Employee> setNameMethodReference = Employee::setName; // NOT POSSIBLE
new *
Consumer<Employee> annonymousSetDefaultName = new Consumer<Employee>() {
    new *
    @Override
    public void accept(Employee employee) {
        employee.setName("default");
    }
};

// Question 3
BiConsumer<Employee, String> setStringName = (Employee e, String name) -> e.setName(name);
BiConsumer<Employee, String> setStringNameMReference = Employee::setName;
new *
BiConsumer<Employee, String> setStringNameAClass = new BiConsumer<Employee, String>() {
    new *
    @Override
    public void accept(Employee employee, String s) {
        employee.setName(s);
    }
};

// Question 4
BiFunction<Integer, Integer, Double> lamdaMathPow = (Integer i1, Integer i2) -> Math.pow(i1, i2);
BiFunction<Integer, Integer, Double> methodReferenceMathPow = Math::pow;
new *
BiFunction<Integer, Integer, Double> annonymous = new BiFunction<Integer, Integer, Double>() {
    new *
    @Override
    public Double apply(Integer integer, Integer integer2) {
        return Math.pow(integer, integer2);
    }
};
```



```java
// Question 5 FIRST POSSIBILITY
Comparator<String> comparator1 = (String s1, String s2) -> s1.compareTo(s2); // FIRST POSSIBILITY
Comparator<String> stringComparator = String::compareTo;
new *
Comparator<String> annonymousComparator = new Comparator<String>() {
    new *
    @Override
    public int compare(String o1, String o2) {
        return o1.compareTo(o2);
    }
};

// Question 5 SECOND POSSIBILITY
BiFunction<String, String, Integer> comparator2 = (String s1, String s2) -> s1.compareTo(s2); // SECOND POSSIBILITY
BiFunction<String, String, Integer> methodReferenceComparator2 = String::compareTo;
new *
BiFunction<String, String, Integer> annonymousComparator2 = new BiFunction<String, String, Integer>() {
    new *
    @Override
    public Integer apply(String s1, String s2) {
        return s1.compareTo(s2);
    }
};
```



```java
// Question 6 FIRST POSSIBILITY
EmployeeComparator comp = new EmployeeComparator();
Comparator<Employee> employeeComparator = (Employee e1, Employee e2) -> comp.compare(e1, e2);
Comparator<Employee> employeeComparatorMethodReference = EmployeeComparator::compare;
new *
Comparator<Employee> annonymouys = new Comparator<Employee>() {
    new *
    @Override
    public int compare(Employee o1, Employee o2) {
        return comp.compare(o1, o2);
    }
};


// Question 6 SECOND POSSIBILITY
EmployeeComparator comp1 = new EmployeeComparator();
BiFunction<Employee, Employee, Integer> employeeComparator1 = (Employee e1, Employee e2) -> comp1.compare(e1, e2);
BiFunction<Employee, Employee, Integer> employeeComparatorMethodReference1 = EmployeeComparator::compare;
new *
BiFunction<Employee, Employee, Integer> annonymouys1 = new BiFunction<>() {
    new *
    @Override
    public Integer apply(Employee employee, Employee employee2) {
        return comp1.compare(employee, employee2);

    }
};
```