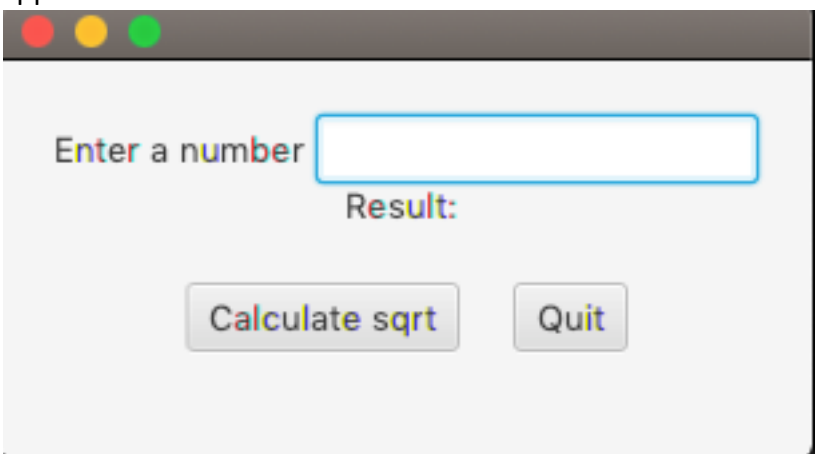
Theoretical questions
Explain each term and give an example for each question (where you can)

1) What is an inner class?
2) What is an anonymous inner class?
3) What is a lambda expression?
4) What does MVC stand for?
5) What is event-driven programming?
6) What are events and events sources?
7) What is an event handler?
8) What is an observable object?
9) What is an abstract class?
10) What is Cloneable interface?
11) What is Comparable interface?
12) What is an interface?
13) What is property binding?
14) What is a generic class?
15) What is type erasure?
16) What is wildcard generic types?
17) What are raw types and backward compatibility (in terms of Generics)?
18) What are some restrictions on generics?

JavaFX

Write a JavaFX application that displays a label together with a text field as shown below. When "Calculate sqrt" button is clicked, get the value of text field, calculate the square root, and display the result right below the text field. When "Quit" button is clicked, exit the application.



Generics
Create a class called AssociationList of generic type <T, E>. This class should serve to store keys and values associated with these keys.

1) This class should contain a method ***void insert (T key, E value)*** which stores a value associated with a key. Make sure that no duplicate keys can be entered. If key already exists, overwrite his associated value with the new value.

2) Build a method ***E retrieve (T key)*** that returns a value that is associated with the given key
3) Build a method ***ArrayList<T> getKeys()*** that returns an array list of keys that you have stored inside your class.

Your testing class should look like this:

```
public class TestAssociation {
    public static void main(String[] args) {
        AssociationList<String, String> associationList = new
AssociationList<>();
        associationList.insert("name", "Anonymous");
        associationList.insert("age", "88");
        associationList.insert("address", "KGB");
        // make sure that a key is inserted only once
        associationList.insert("address", "Mossad");
        System.out.println("Name inside association list is: " +
associationList.retrieve("name"));
        System.out.println("Age inside association list is: " +
associationList.retrieve("age"));
        System.out.println("Address inside association list is: " +
associationList.retrieve("address")); // should be Mossad not KGB
        print(associationList);
    }
    public static<T, E> void print (AssociationList<T, E> list) {
        for(T key: list.getKeys())
            System.out.println(list.retrieve(key));
    }
}
```

Hint: Use two private array lists, ***ArrayList<T> keys*** and ***ArrayList<E> values.*** Every time you want to insert a key value pair, insert into keys array list and values array list. If you need to retrieve a value based on a key, find first the index of the given key in the keys array list, and use this index to access the value in the values array list.
Feel free to solve it in any other way if we can retrieve values based on keys.

Abstract classes and interfaces

Find the output of the given code.
Try to figure it out on your own first, then run it.

```
public class FirstClass {

    public static void main(String [] args) throws Exception {
        FirstAbstractClass[] array = new FirstAbstractClass[] {
                new ThirdClass(),
                new SecondClass(),
                (new FirstAbstractClass() {
                    void foo() {
                        System.out.println("Hello from an Anonymous
class");
                    }
        })};
        for(FirstAbstractClass x: array)
        {
            x.foo();
        }
    }

    private static class SecondClass extends FirstAbstractClass {
        static int a = 11;
        void foo() {

            System.out.println("a in Second class is "+ SecondClass.a);
        }
    }

}
interface FirstInterface {
    int A = 0;
}
abstract class FirstAbstractClass {
    public FirstAbstractClass() {
        System.out.println("Hello from constructor");
    }
    abstract void foo();
    protected static int a = 1;
}
class ThirdClass extends FirstAbstractClass implements FirstInterface {
    public ThirdClass() {
        System.out.println("Hello from Third class constructor");
    }
    void foo() {
        if (A != 0) {
            System.out.println("Hello from Third class");
        }
        if (a != 0) {
            System.out.println("Hello from Third class");
        }
    }
}
```

If you have questions, send an email!