

CONTRACTS

Question: Equality / Contracts BeDifferent[hard]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3     public static void main(String [] arg){
4         String s1=("hi" + null);
5         String s2=("hi" + null);
6         assert s1!=s2;
7     }
8 }
9
```

1 | "hi" + null

SWEN221 2021 T1

Question: Equality / Contracts Boolean1[easy]

Please answer the following question:

```
1 //The answer must be either true or false
2 class Point { int x; int y;
3     public Point(int x, int y) {this.x = x; this.y = y;}
4 }
5 public class Exercise{
6
7     public static void main(String [] arg){
8         Point p1 = new Point(1,2);
9         Point p2 = p1;
10        p2.x = 2;
11        if([???]) { assert p1==p2;}
12        else {assert p1!=p2;}
13    }
14 }
15
```

1 | p1.x == 2

SWEN221 2021 T1

Question: Equality / Contracts Boolean2[easy]

Please answer the following question:

```
1 //The answer must be either true or false
2 class Point { int x; int y;
3     public Point(int x, int y) {this.x = x; this.y = y;}
4 }
5 public class Exercise{
6
7     public static void main(String [] arg){
8         Point p1 = new Point(1,2);
9         Point p2 = new Point(1,2);
10        p2.x = 2;
11        if([???]) { assert p1==p2;}
12        else {assert p1!=p2;}
13    }
14 }
15
```

1 | p1.x == 2

Go back

SWEN221 2021 T1

Question: Equality / Contracts Boolean3[easy]

Please answer the following question:

```
1 //The answer must be either true or false
2 class Point { int x; int y;
3     public Point(int x, int y) {this.x = x; this.y = y;}
4 }
5 public class Exercise{
6
7     public static void main(String [] arg){
8         Point p1 = new Point(1,2);
9         Point p2 = new Point(1,2);
10        if([???]) { assert p1==p2;}
11        else {assert p1!=p2;}
12    }
13 }
14 }
```

1 p1 == p2

[Go back](#)

SWEN221 2021 T1

Question: Equality / Contracts Boolean4[easy]

Please answer the following question:

```
1 //The answer must be either true or false
2 class Point { int x; int y;
3     public Point(int x, int y) {this.x = x; this.y = y;}
4 }
5 public class Exercise{
6
7     public static void main(String [] arg){
8         Point p1 = new Point(1,2);
9         Point p2 = new Point(1,2);
10        if([???]) { assert p1.equals(p2);}
11        else {assert !p1.equals(p2);}
12    }
13 }
14 }
```

1 p1.equals(p2)

SWEN221 2021 T1

Question: Equality / Contracts Equals1[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class Point { int x; int y;
3     public Point(int x, int y) {this.x = x; this.y = y;}
4     [??]
5 }
6 public class Exercise{
7
8     public static void main(String [] arg){
9         Point p1 = new Point(1,2);
10        Point p2 = new Point(1,2);
11        assert p1.equals(p2);
12    }
13 }
14
```

[Go back](#)

```
1 public boolean equals( Object obj )
2 {
3     if ( obj instanceof Point )
4     {
5         Point point = (Point) obj ;
6         return point.x == x && point.y == y ;
7     }
8     else
9     {
10        return false ;
11    }
12 }
```

SWEN221 2021 T1

Question: Equality / Contracts Equals3[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class Point { int x; int y;
3     public Point(int x, int y) {this.x = x; this.y = y;}
4     [??]
5 }
6 public class Exercise{
7
8     public static void main(String [] arg){
9         Point p1 = new Point(1,2);
10        Point p2 = new Point(1,2);
11        assert p1.equals(p2);
12        p2.x=10;
13        assert !p1.equals(p2);
14        assert !p1.equals(new Object());
15        assert !p1.equals(new String());
16        assert !p1.equals(Point.class);
17    }
18 }
19
```

[Go back](#)

```
1 public boolean equals( Object obj )
2 {
3     if ( obj instanceof Point )
4     {
5         Point point = (Point) obj ;
6         return point.x == x && point.y == y ;
7     }
8     else
9     {
10        return false ;
11    }
12 }
```

ddd

SWEN221 2021 T1

Question: Equality / Contracts Equals3[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class Point { int x; int y;
3     public Point(int x, int y) {this.x = x; this.y = y;}
4     [??]
5 }
6 public class Exercise{
7
8     public static void main(String [] arg){
9         Point p1 = new Point(1,2);
10        Point p2 = new Point(1,2);
11        assert p1.equals(p2);
12        p2.x=10;
13        assert !p1.equals(p2);
14        assert !p1.equals(new Object());
15        assert !p1.equals(new String());
16        assert !p1.equals(Point.class);
17    }
18 }
19 }
```

[Go back](#)

```
1 public boolean equals( Object obj )
2 {
3     if ( obj instanceof Point )
4     {
5         Point point = (Point) obj ;
6         return point.x == x && point.y == y ;
7     }
8     else
9     {
10        return false ;
11    }
12 }
```

TIME_1

```
1 //The answer must have balanced parentheses
2 //Complete class Time24 adding add1Hour()
3 class Time24 {
4     //toString>equals and hashCode as generated by eclipse
5     /*omitted*/
6     final int seconds;
7     final int minutes;
8     final int hours;
9     Time24(int seconds, int minutes, int hours){
10         this.seconds=limit60(seconds);
11         this.minutes = limit60(minutes);
12         this.hours=limitHours(hours);
13     }
14     private static int limit60(int v){// 0..59 ok
15         if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "
16             return v;
17     }
18     private static int limitHours(int v){//0..23 ok
19         if(v<0|| v>23){throw new IllegalArgumentException("limitHours ove
20             return v;
21     }
22     static final Time24 twelve= new Time24(0,0,12);
23     [??]
24 }
25
26 public class Exercise{
27     public static void checkAdd(Time24 t){
28         Time24 test=t;
29         for(int i=0;i<24;i++){test=test.add1Hour();}
30         assert test.equals(t);
31         "if I add 24 hours, is the same as before for:" +t;
32     }
33     public static void main(String[]arg){
34         checkAdd(Time24.twelve);
35         checkAdd(new Time24(10,10,10));
36         checkAdd(new Time24(59,59,23));
37     }
38 }
```

TIME_2

```
1 public Time24 add1Hour() {
2     return new Time24(seconds, minutes, limitHours(hours));
3 }
4
5 public Time24 remove1Hour() {
6     return new Time24(seconds, minutes, 24-limitHours(hours));
7 }
```

```

3 //Complete class Time24 adding add1Hour(), remove1Hour()
4 class Time24 {
5     //toString,equals and hashCode as generated by eclipse
6     /*omitted*/
7     final int seconds;
8     final int minutes;
9     final int hours;
10    Time24(int seconds, int minutes, int hours){
11        this.seconds=limit60(seconds);
12        this.minutes = limit60(minutes);
13        this.hours=limitHours(hours);
14    }
15    private static int limit60(int v){// 0..59 ok
16        if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "+v);
17        return v;
18    }
19    private static int limitHours(int v){//0..23 ok
20        if(v<0|| v>23){throw new IllegalArgumentException("limitHours over "+v);
21        return v;
22    }
23    static final Time24 twelve= new Time24(0,0,12);
24    [???
25    }
26
27    public class Exercise{
28        public static void checkAddRemove(Time24 t,int howMuch){
29            Time24 test=t;
30            for(int i=0;i<howMuch;i++){test=test.add1Hour();}
31            for(int i=0;i<howMuch;i++){test=test.remove1Hour();}
32            assert test.equals(t);
33            "for hours: if I add "+howMuch+" and I remove "+howMuch+" is the
34        }
35        public static void checkAdd(Time24 t){
36            {Time24 test=t;
37            for(int i=0;i<24;i++){test=test.add1Hour();}
38            assert test.equals(t);
39            "if I add 24 hours, is the same as before:"+t;
40        }
41        public static void checkRemove(Time24 t){//as for checkAdd
42        /*omitted*/
43    }
44    public static void checkAll(Time24 t){
45        checkAdd(t);

```

```

public Time24 add1Hour() {
    return new Time24(seconds,minutes,limitHours(hours));
}

public Time24 remove1Hour(){
    return new Time24(seconds,minutes,24-limitHours(hours));
}

public Time24 add1Minute(){
    if(minutes==1440){return twelve;}
    return new Time24(seconds,limit60(minutes%60),
                      limitHours(minutes/60));
}

public Time24 remove1Minute(){
    return new Time24(seconds,60-limit60(minutes%60),
                      limitHours(hours-minutes/60));
}

```

TIME_3

```

1 //The answer must have balanced parentheses
2 //HINT: Reuse the code of the former question!
3 //Complete class Time24 adding add1Hour(), remove1Hour()
4 //add1Minute(), remove1Minute()
5 class Time24 {
6     //toString,equals and hashCode as generated by eclipse
7     /*omitted*/
8     final int seconds;
9     final int minutes;
10    final int hours;
11    Time24(int seconds, int minutes, int hours){
12        this.seconds=limit60(seconds);
13        this.minutes = limit60(minutes);
14        this.hours=limitHours(hours);
15    }
16    private static int limit60(int v){// 0..59 ok
17        if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "+v);
18        return v;
19    }
20    private static int limitHours(int v){//0..23 ok
21        if(v<0|| v>23){throw new IllegalArgumentException("limitHours over "+v);
22        return v;
23    }
24    static final Time24 twelve= new Time24(0,0,12);
25    [???
26    }
27
28    public class Exercise{
29        /*omitted*/
30        public static void main(String[]arg){
31            /*omitted*/
32        }
33    }

```

```

1    public Time24 add1Hour() {
2        int updatedHour = hours + 1;
3        if (updatedHour >= 24) {
4            updatedHour %= 24;
5        }
6        return new Time24(seconds, minutes, updatedHour);
7    }
8
9    public Time24 remove1Hour() {
10        int updatedHour = hours - 1;
11        if (updatedHour < 0) {
12            updatedHour += 24;
13        }
14        return new Time24(seconds, minutes, updatedHour);
15    }
16
17    public Time24 add1Minute() {
18        int updatedMinutes = minutes + 1;
19        int updatedHour = hours;
20        if (updatedMinutes >= 60) {
21            updatedMinutes %= 60;
22            updatedHour = hours + 1;
23            if (updatedHour >= 24) {
24                updatedHour %= 24;
25            }
26        }
27        return new Time24(seconds, updatedMinutes, updatedHour);
28    }
29
30    public Time24 remove1Minute() {
31        int updatedMinutes = minutes - 1;
32        int updatedHour = hours;
33        if (updatedMinutes < 0) {
34            updatedMinutes += 60;
35            updatedHour = hours - 1;
36            if (updatedHour < 0) {
37                updatedHour += 24;
38            }
39        }
40        return new Time24(seconds, updatedMinutes, updatedHour);
41    }

```

TIME_4

```
1 //The answer must have balanced parentheses
2 //HINT: Reuse the code of the former question!
3 //Complete class Time24 adding add1Hour(), remove1Hour()
4 //add1Minute(), remove1Minute(), add1Second(), remove1Second()
5 class Time24 {
6     //toString>equals and hashCode as generated by eclipse
7     /*omitted*/
8     final int seconds;
9     final int minutes;
10    final int hours;
11    Time24(int seconds, int minutes, int hours){
12        this.seconds=limit60(seconds);
13        this.minutes = limit60(minutes);
14        this.hours=limitHours(hours);
15    }
16    private static int limit60(int v){// 0..59 ok
17        if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "+v)}
18        return v;
19    }
20    private static int limitHours(int v){//0..23 ok
21        if(v<0|| v>23){throw new IllegalArgumentException("limitHours over "+v)}
22        return v;
23    }
24    static final Time24 twelve= new Time24(0,0,12);
25    [???]
26}
27
28 public class Exercise{
29     /*omitted*/
30     public static void main(String[]arg){
31         /*omitted*/
32     }
33 }
```

TIME_5

```
1 //The answer must have balanced parentheses
2 //HINT: Reuse the code of the former question!
3 //Complete class Time12 adding add1Hour(), remove1Hour()
4 //add1Minute(), remove1Minute(), add1Second(), remove1Second()
5 class Time12 {
6     //toString,equals and hashCode as generated by eclipse
7     /*omitted*/
8     final int seconds;
9     final int minutes;
10    final int hours;
11    final boolean isPm;
12    Time12(int seconds, int minutes, int hours,boolean isPm){
13        this.seconds=limit60(seconds);
14        this.minutes = limit60(minutes);
15        this.hours=limitHours(hours);
16        this.isPm=isPm;
17    }
18    private static int limit60(int v){// 0..59 ok
19        if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "+v)
20            return v;}
21    private static int limitHours(int v){//1..12 ok
22        if(v<1|| v>12){throw new IllegalArgumentException("limitHours over "
23            return v;}
24    static final Time12 twelve= new Time12(0,0,12,true);
25    [???]
26}
27    public class Exercise{
28        /*omitted*/
29    }
30    public static void main(String[]arg){
31        /*omitted*/
32    }
33}
34
```

S

EXCEPTIONS:

SWEN221 2021 T1

Question: Exceptions After Try [medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class EmptyArrayException extends Exception{
3     class ArrayUtil{
4         public static Object lastElement (Object[] array) throws EmptyArrayException{
5             if(array.length>0) return array[array.length-1];
6             throw new EmptyArrayException();
7         }
8     }
9     public class Exercise{
10        public static void test(){
11            try{
12                assert ArrayUtil.lastElement(new Integer[]{1,2,3}).equals(3);
13                assert ArrayUtil.lastElement(new Integer[]{1,2}).equals(2);
14                assert ArrayUtil.lastElement(new Integer[]{1}).equals(1);
15                assert ArrayUtil.lastElement(new Integer[]{}).equals(1);
16            }
17            [??]
18            assert false:"Code not reachable";
19        }
20        public static void main(String [] arg){
21            try{ test(); }
22            catch(Throwable t){/*the test suit
23                logs t on the test results:
24                a single place for error logging.*/
25        }
26    }
27 }
```

[Go back](#)

S

Enter your code here:

```
1 catch (Exception e) {
2
3 }
```

[SEND](#)

[CLEAR](#)

SWEN221 2021 T1

Question: Exceptions Declaring1 [easy]

Please answer the following question:

```
1 //The answer must have no { and }
2 class EmptyArrayException extends [??]{
3     class ArrayUtil{
4         public static Object lastElement (Object[] array){
5             if(array.length>0) return array[array.length-1];
6             throw new EmptyArrayException();
7         }
8     }
9     public class Exercise{
10        public static void main(String [] arg){
11            assert ArrayUtil.lastElement(new Integer[]{1,2,3}).equals(3);
12            assert ArrayUtil.lastElement(new Integer[]{1,2}).equals(2);
13            assert ArrayUtil.lastElement(new Integer[]{1}).equals(1);
14        }
15    }
16 }
```

[Go back](#)

S

Enter your code here:

```
1 RuntimeException
```

[SEND](#)

[CLEAR](#)

SWEN221 2021 T1

Question: Exceptions Declaring3[medium]

Please answer the following question:

```
1 //The answer must be the shortest possible
2 class EmptyArrayException extends RuntimeException{
3     class ArrayUtil{
4         public static Object lastElement (Object[] array){???
5             if(array.length>0) return array[array.length-1];
6             throw new EmptyArrayException();
7         }
8     }
9     public class Exercise{
10         public static void test(){
11             assert ArrayUtil.lastElement(new Integer[]{1,2,3}).equals(3);
12             assert ArrayUtil.lastElement(new Integer[]{1,2}).equals(2);
13             assert ArrayUtil.lastElement(new Integer[]{1}).equals(1);
14             assert ArrayUtil.lastElement(new Integer[]{}).equals(1);
15             assert false:"Code not reachable";
16         }
17         public static void main(String [] arg){
18             try{ test();}
19             catch(Throwable t){/*the test suite
20                 logs t on the test results:
21                 a single place for error logging.*/
22         }
23     }
24 }
```

[Go back](#)

Enter your code here:

```
1 throws EmptyArrayException
```

[SEND](#) [CLEAR](#)

S

SWEN221 2021 T1

Question: Exceptions GoodToKnow[hard]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3     public static int a()throws Throwable{
4         return 1;
5     }
6     public static int b(){
7         try{ return b();}
8         catch(Throwable t){throw new Error(t);}
9         [??]
10    }
11    public static void main(String [] arg){
12        assert b()==0;
13    }
14 }
```

Enter your code here:

```
1 finally {
2     return 0;
3 }
```

[SEND](#) [CLEAR](#)

S

Question: Exceptions MakeADifference[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3
4     public static int m1(){
5         try{
6             try{if(true){throw [???]}}
7             catch(Error e){return 2;}
8             finally{return 2;}
9         }
10        catch(Throwable t){return 10;}
11    }
12    public static int m2(){
13        try{
14            try{if(true){throw [???]}}
15            catch(Error e){return 2;}
16            return 2;
17        }
18        catch(Throwable t){return 10;}
19    }
20    public static void main(String [] arg){
21        assert (m1() != m2()):"assertion:"+m1()+"!="+m2();
22    }
23}
24
```

S

Question: Exceptions Person1[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 //Declare a class Person with fields String name and int age.
3 //Declare a constructor initializing the fields and
4 //Override the method toString to produce the required results.
5 import java.util.HashMap;
6 import java.util.List;
7 import java.util.ArrayList;
8 import java.util.Arrays;
9
10 class Person {
11     String name ;
12     int age ;
13
14     public Person(String name, int age) {
15         super();
16         this.name = name;
17         this.age = age;
18     }
19
20     @Override
21     public String toString() {
22         return name + ":" + age ;
23     }
24 }
25
26 public class Exercise{
27     public static void main(String [] arg){
28         assert "Marco:33".equals(new Person("Marco",33).toString());
29         assert "Adam:13".equals(new Person("Adam",13).toString());
30         assert "Godzilla:450000000".equals(new Person("Godzilla",450000000));
31         assert "Lidia:83".equals(new Person("Lidia",83).toString());
32     }
33 }
34
```

S

Enter your code here:

```
1 new IllegalArgumentException("ERROR");
```

Enter your code here:

```
1 class Person {
2     String name ;
3     int age ;
4
5     public Person(String name, int age) {
6         super();
7         this.name = name;
8         this.age = age;
9     }
10
11     @Override
12     public String toString() {
13         return name + ":" + age ;
14     }
15 }
```

SEND

CLEAR

Question: Exceptions Person2[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 //As for the former question, declare a class Person with fields String
3 //Declare a constructor initializing the fields and
4 //Override the method toString to produce the required results.
5 //In addition: check as an invariant that the age can not
6 //be negative, and the name can not be the empty string.
7 //In case this invariant is violated, throw IllegalArgumentException
8 //import java.util.HashMap;
9 import java.util.List;
10 import java.util.ArrayList;
11 import java.util.Arrays;
12 [??]
13 public class Exercise{
14     public static void main(String [] arg){
15         assert "Marco:33".equals(new Person("Marco",33).toString());
16         assert "Adam:13".equals(new Person("Adam",13).toString());
17         assert "Godzilla:45000000".equals(new Person("Godzilla",45000000));
18         assert "Lidia:83".equals(new Person("Lidia",83).toString());
19         try{new Person("Lidia",-1);assert false;}catch(IllegalArgumentException e)
20         try{new Person("Adam",-100);assert false;}catch(IllegalArgumentException e)
21         try{new Person("",100);assert false;}catch(IllegalArgumentException e)
22         try{new Person("",10);assert false;}catch(IllegalArgumentException e)
23         try{new Person("",-100);assert false;}catch(IllegalArgumentException e)
24     }
25 }
26 }
```

```
1 class Person {
2     String name;
3     int age;
4
5     public Person(String n,int a){
6         setName(n);
7         setAge(a);
8     }
9
10    public void setName(String name) throws IllegalArgumentException
11    {
12        if(name=="")
13            throw new IllegalArgumentException("Name cannot be an empty string");
14        else
15            this.name=name;
16    }
17
18    public void setAge(int age) throws IllegalArgumentException
19    {
20        if(age<0)
21            throw new IllegalArgumentException("Age cannot be negative");
22        else
23            this.age=age;
24    }
25
26    @Override
27    public String toString() {
28        return name + ":" + age;
29    }
30 }
```

S

```
//The answer must be the shortest possible
class EmptyArrayException extends RuntimeException{}
class ArrayUtil{
    public static Object lastElement (Object[] array)[???]{}
        if(array.length>0)return array[array.length-1];
        throw new EmptyArrayException();
    }
}
public class Exercise{
    public static void test(){
        assert ArrayUtil.lastElement(new Integer[]{1,2,3}).equals(3);
        assert ArrayUtil.lastElement(new Integer[]{1,2}).equals(2);
        assert ArrayUtil.lastElement(new Integer[]{1}).equals(1);
        assert ArrayUtil.lastElement(new Integer[]{}).equals(1);
        assert false:"Code not reachable";
    }
    public static void main(String [] arg){
        try{ test();}
        catch(Throwable t){/*the test suit
            logs t on the test results:
            a single place for error logging.**/}
    }
}
```

S

Question: Generics Generic1[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 class Box<[??] > {
3     private [??] contents;
4
5     public Box([??] contents) {
6         this.contents = contents;
7     }
8
9     public [??] getContents() {
10        return contents;
11    }
12 }
13
14 public class Exercise {
15     public static void main(String[] args) {
16         Box<Integer> intBox = new Box<Integer>(1);
17         assert intBox.getContents().equals(1);
18         Box<String> stringBox = new Box<String>("Dave");
19         assert stringBox.getContents().equals("Dave");
20     }
21 }
22
```

[Go back](#)

Enter your code here:

1	T
---	---

SEND **CLEAR**

S

Question: Generics Generic2[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 class NumberBox<T [??] Number> {
3     private T contents;
4
5     public NumberBox(T contents) {
6         this.contents = contents;
7     }
8
9     public T getContents() {
10        return contents;
11    }
12 }
13
14 public class Exercise {
15     public static void main(String[] args) {
16         NumberBox<Integer> intBox = new NumberBox<Integer>(1);
17         assert intBox.getContents().equals(1);
18         NumberBox<Float> floatBox = new NumberBox<Float>(3.145f);
19         assert floatBox.getContents().equals(3.145f);
20     }
21 }
```

S

Enter your code here:

1	extends
---	---------

SEND **CLEAR**

Question: Generics Generic3[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 class Pair [???] {
3     private S first;
4     private T second;
5
6     public Pair(S first, T second) {
7         this.first = first;
8         this.second = second;
9     }
10
11    public S getFirst() { return first; }
12
13    public T getSecond() { return second; }
14
15
16    public class Exercise {
17        public static void main(String[] args) {
18            Pair<String, Integer> p1 = new Pair<String, Integer>("Hello", 1);
19            assert p1.getFirst().equals("Hello");
20            assert p1.getSecond().equals(1);
21        }
22    }
23 }
```

Enter your code here:

```
1 | S, T >
```

S

Question: Generics Generic4[hard]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 import java.util.*;
3 public class Exercise {
4
5     static String get(List<? [???] > strings, int element) {
6         return strings.get(element);
7     }
8
9     public static void main(String[] args) {
10         ArrayList<String> strings = new ArrayList<String>();
11         strings.add("Hello");
12         strings.add("Dave");
13         assert get(strings,0).equals("Hello");
14         assert get(strings,1).equals("Dave");
15     }
16 }
17 }
```

Enter your code here:

```
1 | extends String
```

S

Question: Generics Generic5[hard]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 import java.util.*;
3 public class Exercise {
4
5     static [??] String get(List<T> strings, int element) {
6         return strings.get(element);
7     }
8
9     public static void main(String[] args) {
10        ArrayList<String> strings = new ArrayList<String>();
11        strings.add("Hello");
12        strings.add("Dave");
13        assert get(strings,0).equals("Hello");
14        assert get(strings,1).equals("Dave");
15    }
16}
17
```

```
1 <T extends String>
```

S

Question: Generics Generic6[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 import java.util.*;
3
4 interface Father {
5     public boolean isFather();
6 }
7
8 interface Mother {
9     public boolean isMother();
10}
11
12 class Child implements Mother,Father {
13     public boolean isMother() { return false; }
14     public boolean isFather() { return false; }
15 }
16
17 public class Exercise {
18     @SuppressWarnings("unchecked")
19     static <T extends Mother [??]> T f() {
20         return (T) new Child();
21     }
22
23     public static void main(String[] args) {
24         assert f().isMother() == false;
25         assert f().isFather() == false;
26     }
27 }
28
```

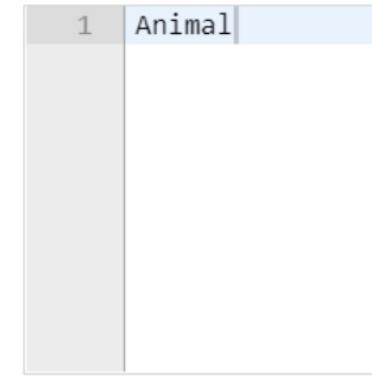
```
1 & Father
```

S

```

1 // The answer must have balanced parentheses
2 import java.util.ArrayList;
3 import java.util.List;
4
5 interface Animal { }
6
7 class Cat implements Animal {
8     public String toString() {
9         return "Cat";
10    }
11 }
12
13 class Dog implements Animal {
14     public String toString() {
15         return "Dog";
16    }
17 }
18
19 public class Exercise {
20
21     public static List<[??]> createList(String... names) {
22         ArrayList<[??]> r = new ArrayList<[??]>();
23         for(String name : names) {
24             switch(name) {
25                 case "Dog":
26                     r.add(new Dog());
27                     break;
28                 case "Cat":
29                     r.add(new Cat());
30                     break;
31             }
32         }
33         return r;
34     }
35
36     public static void main(String[] args) {
37         assert createList("Cat").toString().equals("[Cat]");
38         assert createList("Dog").toString().equals("[Dog]");
39         assert createList("Cat", "Dog").toString().equals("[Cat, Dog]");
40     }
41 }
42

```



S

```

1 // The answer must have balanced parentheses
2 import java.util.*;
3
4 class Parent {
5     public List<? [??] Child> get() {
6         ArrayList<Parent> r = new ArrayList<Parent>();
7         r.add(this);
8         return r;
9     }
10    public String toString() { return "Parent"; }
11 }
12
13 class Child extends Parent {
14     public List<? [??] Parent> get() {
15         ArrayList<Object> r = new ArrayList<Object>();
16         r.add(this);
17         return r;
18     }
19     public String toString() { return "Child"; }
20 }
21
22 public class Exercise {
23
24     public static void main(String[] args) {
25         Parent parent = new Parent();
26         Child child = new Child();
27         assert parent.get().toString().equals("[Parent]");
28         assert child.get().toString().equals("[Child]");
29     }
30 }
31

```



S

Question: Generics ApplicableMethod[hard]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 import java.util.*;
3 public class Exercise{
4     public static void main(String [] arg){
5         List<String> sl = new ArrayList<String>();
6         List<Integer> il = new ArrayList<Integer>();
7         sl.add("foo");sl.add("bar");
8         il.add(1);il.add(2);
9         assert myToString(sl).equals("foobar");
10        assert myToString(il).equals("12");
11    }
12    static String myToString [???]
13 }
14
```

```
1 (List<? extends Object> sl) {
2     StringBuilder sb = new StringBuilder();
3
4     for(Object a:sl){
5         sb.append(a);
6     }
7     return sb.toString();
8 }
```

S

Question: Generics Maps1[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses and not use "+"
2 //Write a function that takes a map CityName -> Population
3 //and return the name of the most populated city
4 import java.util.HashMap;
5 import java.util.List;
6 import java.util.Arrays;
7 public class Exercise{
8     public static String findMostPopulated(HashMap<String,Integer> that){
9         [??]
10    }
11    public static void check(String expected,List<String>city,List<Integer>
12    HashMap<String,Integer>map=new HashMap<>();
13    for(int i=0;i<city.size();i++){
14        map.put(city.get(i),population.get(i));
15        assert expected.equals(findMostPopulated(map)):"expected "+expected+
16    }
17    public static void main(String [] arg){
18        check("Rome",Arrays.asList("Rome","Milan","Venice"),
19            Arrays.asList(500,100,10));
20        check("Milan",Arrays.asList("Rome","Milan","Venice"),
21            Arrays.asList(100,500,10));
22        check("Milan",Arrays.asList("Milan"),
23            Arrays.asList(100));
24        try{findMostPopulated(new HashMap<String,Integer>());assert false;}c
25    }
26 }
```

```
1 int mostPop = 0;
2 String mostPopCity = "";
3 if(that.isEmpty()){
4     throw new java.util.NoSuchElementException();
5 }
6 for(String key : that.keySet() ){
7     if(that.get(key) >= mostPop){
8         mostPop = that.get(key);
9         mostPopCity = key;
10    }
11 }
12 return mostPopCity;
```

S

```
1 //The answer must have balanced parentheses
2 class Point{ int x;int y;
3     Point(int x, int y){this.x=x;this.y=y;}
4 }
5 class ColPoint extends Point{ int colour;
6     ColPoint(int x, int y,int colour){
7         super(x,y);this.colour=colour;
8     }
9 public class Exercise{
10    public static void main(String [] arg){
11        ColPoint c1 = new ColPoint(1,2,124);
12        ColPoint c2 = new ColPoint(1,3,120);
13        Point c3 = new Point(1,5);
14        assert(c1 == min(c1,c3));
15        c2 = min(c1,c2);
16        assert(c1 == c2);
17    }
18    [??]
19 }
```

```
1 public static ColPoint min(ColPoint c1,Point c2){
2     return c1;
3 }
```

S

```
1 //The answer must have balanced parentheses
2 //Extend class A<T> so that findMax finds
3 //the smallest Integer number in the list
4 import java.util.Arrays;
5 import java.util.List;
6 abstract class A<T>{
7     T findMax(List<T> that){
8         if(that.isEmpty())throw new java.util.NoSuchElementException();
9         T candidate=that.get(0);
10        for(T s:that){
11            candidate=better(candidate,s);
12        }
13        return candidate;
14    }
15    abstract T better(T e1, T e2);
16 }
17
18 class B extends A<Integer>{
19     Integer better(Integer e1, Integer e2){
20         return Math.min(e1, e2);
21     }
22 }
23
24 public class Exercise{
25     static void check(Integer expected,List<Integer> data){
26         Integer result=new B().findMax(data);
27         assert expected.equals(result): "on data: "+data+"expected: "+expected;
28     }
29     public static void main(String [] arg){
30         check(3,Arrays.asList(22,12,13,14,55,102,13,3));
31         check(-12,Arrays.asList(22,-12,13,14,55,102,13,3));
32         check(1,Arrays.asList(1));
33     }
34 }
```

```
1 | return Math.min(e1, e2);
```

```
2 //Extend class A<T> so that findMax finds
3 //the tallest student in the list
4 import java.util.Arrays;
5 import java.util.List;
6 abstract class A<T>{
7     T findMax(List<T> that){
8         if(that.isEmpty()) {throw new java.util.NoSuchElementException();}
9         T candidate=that.get(0);
10        for(T s:that){
11            candidate=better(candidate,s);
12        }
13        return candidate;
14    }
15    abstract T better(T e1, T e2);
16 }
17
18 class Student{
19     final String name;
20     final float averageMark;
21     final int height;
22     Student(String name, float averageMark, int height)
23     {
24         this.name=name;
25         this.averageMark=averageMark;
26         this.height=height;
27     }
28     public String toString(){
29         return "S("+name+","+averageMark+","+height+)";
30     }
31 }
32 class B extends A<Student>{
33     [???]
34 }
35
36 public class Exercise{
37     static void check(String expected, Student ... data)
38     {
39         Student result=new B().findMax(Arrays.asList(data));
40         assert expected.equals(result.name):
41             "on data: "+data+"expected: "+expected+" result
42             "+result.name);
43     }
44     public static void main(String [] arg){
45         check("Paul",
46             new Student("Paul",10,185),
47             new Student("John",12,175),
48             new Student("Sarah",14,190));
49     }
50 }
```

```
1 Student better(Student e1, Student e2) {  
2     if(e1.height<=e2.height){  
3         return e2;  
4     } else {  
5         return e1;  
6     }  
7 }
```

```

1 //The answer must have balanced parentheses
2 //Extend class A<T> so that findMax finds
3 //the student whose name is last in alphabetical order
4 import java.util.Arrays;
5 import java.util.List;
6 abstract class A<T>{
7     T findMax(List<T> that){
8         if(that.isEmpty()){throw new java.util.NoSuchElementException();}
9         T candidate=that.get(0);
10        for(T s:that){
11            | candidate=better(candidate,s);
12        }
13        return candidate;
14    }
15    abstract T better(T e1, T e2);
16 }
17
18 class Student{
19     final String name; final float averageMark; final int height;
20     Student(String name, float averageMark, int height){
21         this.name=name;this.averageMark=averageMark; this.height=height;}
22     public String toString(){return "S("+name+","+averageMark+","+height+"
23     }
24
25 class B extends A<Student>{
26     [???
27 }
28
29 public class Exercise{
30     static Student S(String name, float averageMark, int height){
31         return new Student(name,averageMark,height);}
32     static void check(String expected,List<Student> data){
33         Student result=new B().findMax(data);
34         assert expected.equals(result.name): "on data: "+data+"expected: "+e
35     }
36     public static void main(String [] arg){
37         check("zorro", Arrays.asList(S("Adam",10,185),S("Zorro",8,145)));
38         check("Paul", Arrays.asList(S("Paul",10,185),S("Lan",8,145),S("Marco
39         check("Thor", Arrays.asList(S("Paul",10,185),S("Thor",10,215),S("Mar
40         check("Godzilla", Arrays.asList(S("Godzilla",1,6500)));
41     }
42 }

```

```

1 Student better(Student e1, Student e2) {
2     if(e1.name.compareTo(e2.name)>0){
3         return e1;
4     } else if(e1.name.compareTo(e2.name)<=0){
5         return e2;
6     } else {
7         return e1;
8     }
9 }

```

S

```

1 //The answer must have balanced parentheses
2 //Extend class A<T> so that findMax finds
3 //the student with the highest mark
4 import java.util.Arrays;
5 import java.util.List;
6 abstract class A<T>{
7     T findMax(List<T> that){
8         if(that.isEmpty()){throw new java.util.NoSuchElementException();}
9         T candidate=that.get(0);
10        for(T s:that){
11            | candidate=better(candidate,s);
12        }
13        return candidate;
14    }
15    abstract T better(T e1, T e2);
16 }
17
18 class Student{
19     final String name; final float averageMark; final int height;
20     Student(String name, float averageMark, int height){
21         this.name=name;this.averageMark=averageMark; this.height=height;}
22     public String toString(){return "S("+name+","+averageMark+","+height+"
23     }
24     [???
25
26 public class Exercise{
27     static Student S(String name, float averageMark, int height){
28         return new Student(name,averageMark,height);}
29     static void check(String expected,List<Student> data){
30         Student result=new B().findMax(data);
31         assert expected.equals(result.name): "on data: "+data+"expected: "+e
32     }
33     public static void main(String [] arg){
34         check("Adam", Arrays.asList(S("Adam",10,185),S("Zorro",8,145)));
35         check("Paul", Arrays.asList(S("Paul",10,185),S("Lan",8,145),S("Marco
36         check("Paul", Arrays.asList(S("Paul",10,185),S("Thor",1,215),S("Mar
37         check("Godzilla", Arrays.asList(S("Godzilla",1,6500)));
38     }
39 }
40

```

```

1 class B extends A<Student>{
2     Student better(Student e1, Student e2) {
3         if (e1.averageMark>=e2.averageMark){
4             return e1;
5         } else {
6             return e2;
7         }
8     }
9 }

```

S

```

1 //The answer must have balanced parentheses
2 //Extend class A<T> so that findMax finds
3 //the person having the shortest address
4 // (you have also to define the person class!)
5 import java.util.Arrays;
6 import java.util.List;
7 abstract class A<T>{
8     T findMax(List<T> that){
9         if(that.isEmpty()) {throw new java.util.NoSuchElementException();}
10        T candidate=that.get(0);
11        for(T s:that){
12            candidate=better(candidate,s);
13        }
14        return candidate;
15    }
16    abstract T better(T e1, T e2);
17 }
18 [??]
19
20 public class Exercise{
21     static Person P(String name, String address){
22         return new Person(name,address);
23     }
24     static void check(String expected,List<Person> data){
25         Person result=new B().findMax(data);
26         assert expected.equals(result.name): "on data: "+data+"expected: "+expected;
27     }
28     public static void main(String [] arg){
29         check("Zorro", Arrays.asList(P("Adam","saint monique road 234"),P("Lan","google road 245 California"),P("Thor","under the sea")));
30         check("Lan", Arrays.asList(P("Paul","google road 245 California"),P("Thor","under the sea")));
31     }
32 }

```

S

```

1 //The answer must have balanced parentheses
2 //Extend class A<T> so that findMax finds
3 //the student whose mark is the nearest to
4 //a given target
5 import java.util.Arrays;
6 import java.util.List;
7 abstract class A<T>{
8     T findMax(List<T> that){
9         if(that.isEmpty()) {throw new java.util.NoSuchElementException();}
10        T candidate=that.get(0);
11        for(T s:that){
12            candidate=better(candidate,s);
13        }
14        return candidate;
15    }
16    abstract T better(T e1, T e2);
17 }
18
19 class Student{
20     final String name; final float averageMark; final int height;
21     Student(String name, float averageMark, int height){
22         this.name=name;this.averageMark=averageMark; this.height=height;}
23     public String toString(){return "("+name+","+averageMark+","+height+", "+height+"m)";}
24 }
25 [??]
26
27 public class Exercise{
28     static Student S(String name, float averageMark, int height){
29         return new Student(name,averageMark,height);
30     }
31     static void check(int target,String expected,List<Student> data){
32         Student result=new B(target).findMax(data);
33         assert expected.equals(result.name): "on data: "+data+"expected: "+expected;
34     }
35     public static void main(String [] arg){
36         check(5,"Zorro", Arrays.asList(S("Adam",10,185),S("Zorro",8,145)));
37         check(3,"Lan", Arrays.asList(S("Paul",10,185),S("Lan",2,145),S("Mark",10,185)));
38         check(7,"Thor", Arrays.asList(S("Paul",10,185),S("Thor",7,215),S("Mark",10,185)));
39         check(2,"Godzilla", Arrays.asList(S("Godzilla",1,6500)));
40     }
41 }

```

```

1 class Person {
2     String name,address;
3     public Person(String name, String address) {
4         this.name = name;
5         this.address = address;
6     }
7 }
8
9 class B extends A<Person>{
10
11     @Override
12     Person better(Person e1, Person e2) {
13         if(e1.address.length()<=e2.address.length()){
14             return e1;
15         } else {
16             return e2;
17         }
18     }
19 }

```

```

1 class B extends A<Student>{
2     int target;
3
4     public B(int target) {
5         this.target = target;
6     }
7
8     @Override
9     Student better(Student e1, Student e2) {
10        if(e1.averageMark-target<=e2.averageMark-target){
11            return e1;
12        } else {
13            return e2;
14        }
15    }
16 }

```

INHERITANCE

SWEN221 2021 T1

Question: Inheritance Extends1[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{A(String name){this.name=name;}String name;}
3
4 class B extends A{[???]}
5
6 public class Exercise{
7
8     public static void main(String [] arg){
9         assert (new B().name.equals("Cell"));
10    }
11 }
```

[Go back](#)

```
1 B(){}
2     super("Cell");
3 }
```

SWEN221 2021 T1

Question: Inheritance Extends2[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{A(String name){this.name=name;}String name;}
3
4 class B extends A{[???]}
5
6 public class Exercise{
7
8     public static void main(String [] arg){
9         assert (!new B().name.equals(new B().name));
10    }
11 }
```

[Go back](#)

Enter your code here:

```
1 B(){}
2     super(string.valueOf(Math.random()));
3 }
```

SWEN221 2021 T1

Question: Inheritance Field1[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{
3     private int f=0;
4     public void setF(int f){[???]}
5     public boolean checkF(int expected){return f==expected;}
6 }
7 class B extends A{
8     public void m(){
9         setF(3);
10        assert checkF(3);
11        setF(8);
12        assert checkF(8);
13    }
14 }
15 public class Exercise{
16     public static void main(String [] arg){
17         new B().m();
18     }
19 }
20 }
```

[Go back](#)

```
1 this.f = f;
```

SWEN221 2021 T1

Question: Inheritance Field2[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{
3     private int f=0;
4     public void setF(int _f){f=_f;}
5     public boolean checkF(int expected){return f==expected;}
6 }
7
8 class B extends A{
9     public void setF(int f){[???]}
10    public void m(){
11        setF(3);
12        assert checkF(-3);
13        setF(8);
14        assert checkF(-8);
15    }
16 }
17
18 public class Exercise{
19     public static void main(String [] arg){
20         new B().m();
21     }
22 }
```

[Go back](#)

```
1 super.setF(-f);
```

Question: Inheritance Interface1[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 interface Student {
3     public String getName();
4     public int getID();
5 }
6
7 class MyStudent [???] Student {
8     private String name;
9     private int id;
10
11    public MyStudent(String name, int id) {
12        this.name = name;
13        this.id = id;
14    }
15
16    public String getName() { return name; }
17
18    public int getID() { return id; }
19 }
20
21 public class Exercise {
22     public static void main(String [] arg){
23         Student s = new MyStudent("Dave",300384719);
24         assert s.getName().equals("Dave");
25         assert s.getID() == 300384719;
26     }
27 }
28 }
```

Enter your code here:

1 | implements

SEND **CLEAR**

Question: Inheritance Interface2[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 interface Person {
3     public String getName();
4 }
5
6 interface Student [???] Person {
7     public int getID();
8 }
9
10 class MyStudent implements Student {
11     private String name;
12     private int id;
13
14    public MyStudent(String name, int id) {
15        this.name = name;
16        this.id = id;
17    }
18
19    public String getName() { return name; }
20
21    public int getID() { return id; }
22 }
23
24 public class Exercise {
25     public static void main(String [] arg){
26         Student s = new MyStudent("Dave",300384719);
27         assert s.getName().equals("Dave");
28         assert s.getID() == 300384719;
29     }
30 }
31 }
```

Enter your code here:

1 | extends

Question: Inheritance Interface3[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 class Parent {
3     public int f() { return 123; }
4 }
5
6 class Child extends Parent {
7     public int f() {
8         return [???].f();
9     }
10}
11
12 public class Exercise {
13     public static void main(String[] args) {
14         Child c = new Child();
15         assert c.f() == 123;
16     }
17 }
18
```

Enter your code here:

```
1 new Parent()
```

SWEN221 2021 T1

Question: Inheritance symbolic1[hard]

Please answer the following question:

```
1 //The answer must contains only digits
2 abstract class Parent {
3     int x=100;
4     Parent (int x) {this.x=x;}
5     abstract int m(int x);
6     static Parent k (Parent x) { return x;}
7     int f (int x) { return m(x+1);}
8 }
9
10 class Heir extends Parent {
11     static int x=200;
12     Heir (int x) { super (x+1);}
13     Heir () {this(3);}
14     int m(int x) { return x+2;}
15     int n (int x) { return this.x + x; }
16 }
17
18 public class Exercise{
19     public static void main(String[] arg){
20         Parent p = new Heir();
21         Heir h = new Heir();
22         assert(p.f(h.x)==[???]);
23     }
24 }
```

```
1 203
```

Go back

SWEN221 2021 T1

Question: Inheritance symbolic2[hard]

Please answer the following question:

```
1 //The answer must contains only digits
2 abstract class Parent {
3     int x=100;
4     Parent (int x) {this.x=x;}
5     abstract int m(int x);
6     static Parent k (Parent x) { return x;}
7     int f (int x) { return m(x+1);}
8 }
9
10 class Heir extends Parent {
11     static int x=200;
12     Heir (int x) { super (x+1);}
13     Heir () {this(3);}
14     int m(int x) { return x+2;}
15     int n (int x) { return this.x + x; }
16 }
17
18 public class Exercise {
19     public static void main(String[] arg){
20         Parent p = new Heir();
21         Heir h = new Heir();
22         new Heir().x=5;
23         assert(h.n(p.x)==9);
24     }
25 }
26
```

1 9

[Go back](#)

S

SWEN221 2021 T1

Question: Inheritance symbolic3[hard]

Please answer the following question:

```
1 //The answer must contains only digits
2 abstract class Parent {
3     int x=100;
4     Parent (int x) {this.x=x;}
5     abstract int m(int x);
6     static Parent k (Parent x) { return x;}
7     int f (int x) { return m(x+1);}
8 }
9
10 class Heir extends Parent {
11     static int x=200;
12     Heir (int x) { super (x+1);}
13     Heir () {this(3);}
14     int m(int x) { return x+2;}
15     int n (int x) { return this.x + x; }
16 }
17
18 public class Exercise {
19     public static void main(String[] arg){
20         Parent p = new Heir();
21         Heir h = new Heir();
22         assert((p.k(h)).x==[???]);
23     }
24 }
```

1 4

[Go back](#)

S

MISCELLANEOUS

SWEN221 2021 T1

Question: Misc Concat[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 //write a function concat all the String in a list,
3 //adding "[" at the start and "]" at the end
4 import java.util.ArrayList;
5 import java.util.Arrays;
6 public class Exercise{
7 public static String concat(ArrayList<String> that){
8 [???
9 }
10 public static void main(String [] arg){
11     assert "[abc]".equals(
12         concat(new ArrayList<String>(Arrays.asList("", "a", "b", "c"))));
13     assert "[".equals(concat(new ArrayList<String>())));
14     /*omitted*/
15 }
16 }
```

[Go back](#)

```
1 String concat = "[";
2 for (int i = 0; i < that.size(); ++i) {
3     concat += that.get(i);
4 }
5 return concat += "]";
```

S

SWEN221 2021 T1

Question: Misc IntegerList[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 import java.util.*;
3 public class Exercise{
4 public static void main(String [] arg){
5     List<Integer> elems=new ArrayList<Integer>();
6     elems.add(10);
7     [???
8     elems.add(20);
9     elems.add(30);
10    assert elems.get(0)==10;
11    assert elems.get(1)==50;
12    assert elems.get(2)==70;
13    assert elems.get(3)==20;
14 }
15 }
```

[Go back](#)

```
1 elems.add(50);
2 elems.add(70);
```

S

SWEN221 2021 T1

Question: Misc LongestString[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses and not use "+"
2 //write a function that returns longest string in a list,
3 import java.util.ArrayList;
4 import java.util.Arrays;
5 public class Exercise{
6 public static String longest(ArrayList<String> that){
7     [???
8 }
9 public static void main(String [] arg){
10    assert "foo".equals(
11        longest(new ArrayList<String>(Arrays.asList("", "a", "foo", "b"))));
12    assert "bar".equals(
13        longest(new ArrayList<String>(Arrays.asList("", "bar", "b", "b"))));
14    /*omitted*/
15 }
16 }
```

[Go back](#)

```
1 if (that.size() == 0) {
2     throw new java.util.NoSuchElementException();
3 }
4 String longest = "";
5 int i = 0;
6 for (String s : that) {
7     if (s.length() > i) {
8         i = s.length();
9         longest = s;
10    }
11 }
12 return longest;
```

S

SWEN221 2021 T1

Question: Misc MyAverage[medium]

Please answer the following question:

```
1 //the answer must have balanced parentheses, and not containing "exercise"
2 import java.util.*;
3 public class Exercise{
4     public static float myAverage(List<Float> elems){[?/?]}
5     //compute the average, of on empty lists
6     public static float exerciseAverage(List<Float> elems){
7         /*omitted*/
8     }
9     public static void main(String [] arg){
10        List<List<Float>> tests=Arrays.asList(
11            Arrays.asList(1f,2f,3f,4f,5f,6f,7f),
12            Arrays.asList(-1f,-2f,-3f,-7f,-13f),
13            Arrays.asList(-0.5f,0f,0f,0f,30f,-100f,200f,23f,42f),
14            Arrays.asList(-12f/3f,32f,12f,45f,-1000f,99f),
15            Arrays.asList(-13f/0f,-100f/0f),
16            Arrays.asList(7f,6f,4f,3f,2f,1f,0f,-100f),
17            Arrays.asList(7f,6f,4f,3f,2f,1f,0f,1f,2f,3f,4f,5f,6f,7f,8f)
18        );
19        for(List<Float> ls:tests)
20            assert myAverage(ls)==exerciseAverage(ls): "error on "+ls+" "+myAve
21    }
22 }
```

[Go back](#)

```
1 float sum = 0;
2 for (int i = 0; i < elems.size(); ++i) {
3     sum += elems.get(i);
4 }
5 return (sum / elems.size());|
```

S

SWEN221 2021 T1

Question: Misc MyPow[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses, and not containing "exercise"
2 //exercisePow is an implementation for the pow function
3 public class Exercise{
4     public static int myPow(int base,int exp){[?/?]}
5     public static int exercisePow(int base,int exp){
6         /*omitted*/
7     }
8     public static void main(String [] arg){
9         for(int i=0;i<15;i++)
10             for(int j=0;j<5;j++)
11                 assert myPow(i,j)==exercisePow(i,j): "error on "+ i +"," + j + "
12     }
13 }
```

[Go back](#)

```
1 return (int) Math.pow(base, exp);|
```

S



SWEN221 2021 T1

Question: Misc SingleKeyword[medium]

Please answer the following question:

```
1 //The answer must be a single keyword
2 class A{
3     [??] void foo(String s){assert false;}
4     void foo(Object o){}
5 }
6 public class Exercise{
7     public static void main(String [] arg){
8         new A().foo("Hello");
9     }
10}
11
```

1	private

[Go back](#)

S

Question: Misc UseFindMax1[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses and not use "+"
2 //Using the function findMax, remove the bigger element from the list
3 import java.util.ArrayList;
4 import java.util.List;
5 import java.util.Arrays;
6 public class Exercise{
7     public static Float findMax(List<Float> that){
8         if(that.isEmpty()){throw new java.util.NoSuchElementException();}
9         float candidate=that.get(0);
10        for(Float f:that){
11            if(f>candidate){candidate=f;}
12        }
13        return candidate;
14    }
15    public static void removeMax(List<Float> that){
16        [??]
17    }
18    public static void check(List<Float> original,List<Float> expected){
19        original=new ArrayList<Float>(original);
20        expected=new ArrayList<Float>(expected);
21        removeMax(original);
22        assert original.equals(expected);
23    }
24    public static void main(String [] arg){
25        check(Arrays.asList(1f,2f,3f),Arrays.asList(1f,2f));
26        check(Arrays.asList(100f),new ArrayList<Float>());
27        check(Arrays.asList(4f,1f,2f,3f),Arrays.asList(1f,2f,3f));
28        /*omitted*/
29    }
30}
31
```

1	Float max = findMax(that);
2	that.remove(max);
3	return longest;

[Go back](#)

S

Question: Misc UseFindMax2[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses and not use "+"
2 //Using the function findMax, without modifying the original list,
3 //produce a new list that is like the original but do not have the biggest
4 import java.util.ArrayList;
5 import java.util.List;
6 import java.util.Arrays;
7 public class Exercise{
8     public static float findMax(List<Float> that){
9         if(that.isEmpty()){throw new java.util.NoSuchElementException();}
10        float candidate=that.get(0);
11        for(float f:that){
12            if(f>candidate){candidate=f;}
13        }
14        return candidate;
15    }
16    public static List<Float> removeMax(List<Float> that){
17        [??]
18    }
19    public static void check(List<Float> original,List<Float> expected){
20        ArrayList<Float> original2=new ArrayList<Float>(original);
21        expected=new ArrayList<Float>(expected);
22        List<Float> result=removeMax(original);
23        assert original.equals(original2);
24        assert result.equals(expected);
25    }
26    public static void main(String [] arg){
27        check(Arrays.asList(1f,2f,3f),Arrays.asList(1f,2f));
28        check(Arrays.asList(100f),new ArrayList<Float>());
29        check(Arrays.asList(4f,1f,2f,3f),Arrays.asList(1f,2f,3f));
30        /*omitted*/
31    }
32}
33
```

```
1 Float max = findMax(that);
2 ArrayList<Float> newList = new ArrayList<>();
3 newList.addAll(that);
4 newList.remove(max);
5 return newList;
```

POLYMORPHISM

SWEN221 2021 T1

Question: Polymorphism ManyF[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{}
3 class B extends A{}
4 public class Exercise{
5     public static boolean f(A p1, A p2){ return false;}
6     public static boolean f(B p1, A p2){ return true;}
7     public static boolean f(B p1, B p2){ return false;}
8     public static void main(String [] arg){
9         assert f([???]);
10    }
11 }
12
```

[Go back](#)

1 new B(), new A()

S

SWEN221 2021 T1

Question: Polymorphism ManyM1[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{}
3 class B extends A{}
4 public class Exercise{
5     public static boolean m(A p1){ return false;}
6     public static boolean m(String p1){ return true;}
7     public static boolean m(B p1){ return false;}
8     public static void main(String [] arg){
9         assert m([???]);
10    }
11 }
12
```

[Go back](#)

"Ricky"

S

SWEN221 2021 T1

Question: Polymorphism ManyM2[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{}
3 class B extends A{}
4 public class Exercise{
5     public static boolean m(A p1, B p2){ return false;}
6     public static boolean m(A p1, A p2){ return true;}
7     public static boolean m(B p1, B p2){ return false;}
8     public static void main(String [] arg){
9         assert m([???]);
10    }
11 }
12 }
```

[Go back](#)

1 new A(), new A()

S

SWEN221 2021 T1

Question: Polymorphism ManyMF[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{}
3 class B extends A{}
4 public class Exercise{
5     public static boolean m(A p1, B p2){ return false;}
6     public static boolean m(A p1, A p2){ return true;}
7     public static boolean m(B p1, B p2){ return false;}
8     public static boolean f(A p1, A p2){ return false;}
9     public static boolean f(B p1, A p2){ return true;}
10    public static boolean f(B p1, B p2){ return false;}
11    public static void main(String [] arg){
12        assert m([???]);
13        assert f([???]);
14    }
15 }
16 }
```

[Go back](#)

1 new B(), new A()

S

SWEN221 2021 T1

Question: Polymorphism Poly1[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 class Parent {
3     public int f() { return 123; }
4 }
5
6 class Child extends Parent {
7     public int f() {
8         return [???];
9     }
10}
11
12 public class Exercise {
13     public static void main(String[] args) {
14         Parent p = new Child();
15         assert p.f() == 456;
16     }
17 }
```

Go back

1 456

S

SWEN221 2021 T1

Question: Polymorphism Poly2[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 class Parent {
3     public String name() {
4         return "Parent";
5     }
6 }
7
8 class Child extends Parent {
9     public String name() {
10        return "Child";
11    }
12 }
13
14 public class Exercise {
15     public static void main(String[] args) {
16         Parent p = new [???];
17         assert p.name().equals("Parent");
18     }
19 }
```

Go back

Photos - b.PNG



1 Parent()

S

SWEN221 2021 T1

Question: Polymorphism Poly3[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 interface Parent {
3     [??];
4 }
5 class Child implements Parent {
6     public String name(boolean b) {
7         return "Child " + b;
8     }
9
10    public String name(int i) {
11        return "Child " + i;
12    }
13 }
14
15 public class Exercise {
16     public static void main(String[] args) {
17         Parent p = new Child();
18         assert p.name(1).equals("Child 1");
19         assert p.name(true).equals("Child true");
20     }
21 }
22 }
```

1	public String name(boolean b);
2	public String name(int i);

[Go back](#)

S

SWEN221 2021 T1

Question: Polymorphism Poly4[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2 interface Parent {
3     public String name(Object b);
4 }
5
6 class Child implements Parent {
7     public String name(Object b) {
8         return [??];
9     }
10    public String name(String b) {
11        return "Child " + [??];
12    }
13 }
14
15 public class Exercise {
16     public static void main(String[] args) {
17         Parent p = new Child();
18         assert p.name("String").equals("Child String");
19     }
20 }
21 }
```

1	"Child String"
---	----------------

[Go back](#)

S

```
1 // The answer must have balanced parentheses
2 // This is an example of "double dispatch"
3
4 interface Animal {
5     public boolean eats(Animal o);
6     public boolean isEatenBy(Animal o);
7 }
8
9 class Cat implements Animal {
10    public boolean eats(Animal o) {
11        return o.isEatenBy(this);
12    }
13
14    public boolean isEatenBy(Animal o) {
15        [??]
16    }
17 }
18
19 class Dog implements Animal {
20    public boolean eats(Animal o) {
21        return o.isEatenBy(this);
22    }
23
24    public boolean isEatenBy(Animal o) {
25        [??]
26    }
27 }
28
29 public class Exercise {
30    public static void main(String[] args) {
31        Animal a = new Dog();
32        Animal b = new Cat();
33
34        assert a.eats(b);
35        assert !b.eats(b);
36    }
37 }
38
```

```
1 if (o == this) {
2     return false;
3 }
4 return true;
```

S

Question: Polymorphism Poly6[medium]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2
3 class Cat {
4     public boolean isEatenBy(Object o) {
5         return false;
6     }
7
8     [??]
9 }
10
11 class Dog {
12     private boolean isLittle; // is a little dog?
13
14     public Dog(boolean isLittle) {
15         this.isLittle = isLittle;
16     }
17
18     public boolean isLittle() { return isLittle; }
19
20     public boolean eats(Cat c) {
21         return c.isEatenBy(this);
22     }
23 }
24
25 public class Exercise {
26     public static void main(String[] args) {
27         Cat cat = new Cat();
28         Dog lilDog = new Dog(true);
29         Dog bigDog = new Dog(false);
30
31         assert !lilDog.eats(cat);
32         assert bigDog.eats(cat);
33     }
34 }
```

```
1 public boolean isEatenBy(Dog dog) {
2     if (!dog.isLittle()) {
3         return true;
4     }
5     return false;
6 }
```

[Go back](#)

S

SWEN221 2021 T1



Question: Polymorphism Poly7[easy]

Please answer the following question:

```
1 // The answer must have balanced parentheses
2
3 interface Animal {
4     public String getName();
5 }
6
7 class Cat implements Animal {
8     public String getName() {
9         return "Cat";
10    }
11 }
12
13 class Dog implements Animal {
14     public String getName() {
15         return "Dog";
16    }
17 }
18
19 public class Exercise {
20
21     public static String get([??] animal) {
22         return animal.getName();
23     }
24
25     public static void main(String[] args) {
26         Dog dog = new Dog();
27         Cat cat = new Cat();
28         assert get(dog).equals("Dog");
29         assert get(cat).equals("Cat");
30     }
31 }
```

1	Animal

[Go back](#)

S

```
1 // The answer must have balanced parentheses
2 import java.util.*;
3
4 class Cat {
5     protected String name;
6     public Cat(String name) { this.name = name; }
7     public void fight(Cat target, List<String> log) {
8         log.add(name + " claws " + target.name);
9         target.fightBack(this,log);
10    }
11
12     public void fightBack(Cat target, List<String> log) {
13         log.add(name + " claws back " + target.name);
14     }
15 }
16
17 class RoughCat extends Cat {
18     public RoughCat(String name) {
19         super(name);
20     }
21
22     public void fight(Cat target, List<String> log) {
23         log.add(name + " bites " + target.name);
24         target.fightBack(this,log);
25     }
26
27     public void fightBack(Cat target, List<String> log) {
28         log.add(name + " bites back " + target.name);
29     }
30 }
31
32 public class Exercise {
33     public static void main(String[] args) {
34         List<String> log = new ArrayList<String>();
35         Cat jim = new Cat("Jim");
36         Cat bob = new RoughCat("Bob");
37         bob.fight(jim,log);
38         assert log.toString().equals([???]);
39     }
40 }
41
```

1 log.toString()

S

```
1 // The answer must have balanced parentheses
2 import java.util.*;
3
4 class Cat {
5     protected String name;
6     public Cat(String name) { this.name = name; }
7     public void fight(Cat target, List<String> log) {
8         log.add(name + " claws " + target.name);
9         target.fightBack(this,log);
10    }
11    public void fightBack(Cat target, List<String> log) {
12        log.add(name + " claws back " + target.name);
13    }
14 }
15
16 class RoughCat extends Cat {
17     public RoughCat(String name) {
18         super(name);
19     }
20
21     public void fight(Cat target, List<String> log) {
22         log.add(name + " bites " + target.name);
23         target.fightBack(this,log);|
24     }
25
26     public void fightBack(RoughCat target, List<String> log) {
27         log.add(name + " bites back " + target.name);
28     }
29 }
30
31 public class Exercise {
32     public static void main(String[] args) {
33         List<String> log = new ArrayList<String>();
34
35         Cat jim = new Cat("Jim");
36         Cat bob = new RoughCat("Bob");
37         bob.fight(jim,log);
38
39         assert log.toString().equals([???]);
40     }
41 }
42 }
```

1 log.toString()

S

PUZZLERS

SWEN221 2021 T1

Question: Puzzlers IsThisPossible?[hard]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3     public static void main(String [] arg){
4         Integer i=(null);
5         String s=(null);
6         assert (Object)s== (Object)i;
7     }
8 }
9 
```

1 hull

[Go back](#)

S

SWEN221 2021 T1

Question: Puzzlers PlusFour[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3
4     public static int foo=1;
5
6     public static void main(String [] arg){
7         int foo=10;
8         assert (14==(foo = foo + 4)):"assertion14";
9         assert (18==(foo = foo + 4)):"assertion18";
10        assert (22==(foo = foo + 4)):"assertion22";
11        assert (26==(foo = foo + 4)):"assertion26";
12    }
13 }
14 
```

1 (foo = foo + 4)

[Go back](#)

S

SWEN221 2021 T1

Question: Puzzlers Range1[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3
4     public static void main(String [] arg){
5         [???
6             assert (Integer.MAX_VALUE==i);
7     }
8 }
```

```
1 int i = 2147483647;
```

[Go back](#)

S

SWEN221 2021 T1

Question: Puzzlers RangeBlocked[hard]

Please answer the following question:

```
1 //The answer must have balanced parentheses and not use "Integer" "Float"
2 public class Exercise{
3
4     public static void main(String [] arg){
5         [???
6             assert (Integer.MAX_VALUE==i);
7     }
8 }
```

```
1 int ONE = "x".length();
2 int i = -ONE >>> ONE;
```

[Go back](#)

S

Question: Puzzlers SqlInjection[hard]

Please answer the following question:

```
1 //The answer is not required to have balanced parentheses inside string ]
2 public class Exercise{
3
4     public static void main(String [] arg){
5         int foo=10;
6         assert ("Foo".equals("Bar[???]")):"assertionFoo=Bar?";
7     }
8 }
```

```
1 ".substring(0, 0)+"Foo
```

[Go back](#)

S

REFLECTION

SWEN221 2021 T1

Question: Inner/Anon/Reflection ArrayUtil.format1[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class ArrayUtil{
3
4     public static String format(Object[] array, String opening, String closing){
5         String result=opening;
6         if(array.length==0) return result+closing;
7         for(int i=0;i<array.length-1;i++){
8             result+=array[i]+separator;
9         }
10        return result+array[array.length-1]+closing;
11    }
12 }
13 public class Exercise{
14     public static void main(String [] arg){
15         assert "[1, 2, 3]".equals(ArrayUtil.format(new Integer[]{1,2,3},[???]));
16         assert "[4, 5, 6]".equals(ArrayUtil.format(new Integer[]{4,5,6},[???]));
17         assert "[7, 8, 9]".equals(ArrayUtil.format(new Integer[]{7,8,9},[???]));
18     }
19 }
20 }
```

1 "[" , "]" , ", "

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection ArrayUtil.format2[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class ArrayUtil{String opening="["; String closing="]"; String separator
3     public final String format(Object[] array){
4         String result=opening;
5         if(array.length==0) return result+closing;
6         for(int i=0;i<array.length-1;i++){
7             result+=array[i]+separator;
8         }
9         return result+array[array.length-1]+closing;
10    }
11 }
12 public class Exercise{
13     public static void main(String [] arg){
14         ArrayUtil u=[???];
15         assert "[1; 2; 3]".equals(u.format(new Integer[]{1,2,3}));
16         assert "[4; 5; 6]".equals(u.format(new Integer[]{4,5,6}));
17         assert "[7; 8; 9]".equals(u.format(new Integer[]{7,8,9}));
18     }
19 }
20 
```

1 new ArrayUtil(){{super.separator="; "};}}

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection ArrayUtil.format3[hard]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class ArrayUtil{String opening="["; String closing="]"; String separator
3     public final String format(Object[] array){
4         String result=opening;
5         if(array.length==0) return result+closing;
6         for(int i=0;i<array.length-1;i++){
7             result+=array[i]+separator;
8         }
9         return result+array[array.length-1]+closing;
10    }
11 }
12 public class Exercise{
13     public static void main(String [] arg){
14         ArrayUtil u=(new Arrayutil(){[???]});
15         assert "[1; 2; 3]".equals(u.format(new Integer[]{1,2,3}));
16         assert "[4; 5; 6]".equals(u.format(new Integer[]{4,5,6}));
17         assert "[7; 8; 9]".equals(u.format(new Integer[]{7,8,9}));
18     }
19 }
20 
```

1 [super.separator = "; ";]}

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection Collections.sort[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 import java.util.*;
3 class Point{ int x,int y;
4     Point(int x, int y){this.x=x;this.y=y;}
5     public String toString(){return "["+x+","+y+"]";}
6 }
7 public class Exercise{
8
9     public static void main(String [] arg){
10         ArrayList<Point> ps=new ArrayList<Point>();
11         ps.add(new Point(2,2));
12         ps.add(new Point(1,2));
13         ps.add(new Point(3,2));
14         ps.add(new Point(2,5));
15         Collections.sort(ps,[???]);
16         assert(ps.toString().equals("[[1,2], [2,2], [2,5], [3,2]]")):ps.toSt
17     }
18 }
19
```

[Go back](#)

```
1 new Comparator<Point>() {
2     @Override
3     public int compare(Point o1, Point o2) {
4         if (o1.x < o2.x) {
5             return -1;
6         } else if (o1.x == o2.x) {
7             if (o1.y<o2.y) {
8                 return -1;
9             } else if(o1.y == o2.y) {
10                return 0;
11            }
12        } else {
13            return 1;
14        }
15    }
16 }
17 }
```

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection Instance2[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{ int m(){return 1;}}
3 public class Exercise{
4     public static void main(String [] arg){
5         final A a=[???];
6         assert a.m()==2;
7     }
8 }
```

[Go back](#)

```
1 new A(){
2     int m() {
3         return 2;
4     }
5 }
```

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax1[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{
3     static class B{
4         static class C{ B b;
5             C(B b){this.b=b;}
6             int foo(){return 42;}
7         }
8     }
9     public class Exercise{
10        public static void main(String [] arg){
11            assert ([???].foo()==42);
12        }
13    }
14 }
15 }
```

[Go back](#)

```
1 new A.C(new A.B())
```

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax2[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{
3     static class B{
4         static class C{ B b;
5             C(B b){this.b=b;}
6             int foo(){return 42;}
7         }
8     }
9 }
10 public class Exercise{
11
12     public static void main(String [] arg){
13         assert ([???].foo()==42);
14     }
15 }
16 }
```

[Go back](#)

```
1 new A.B.C(new A.B())
```

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax3[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{
3     private static class B{}
4     static class C{ B b;
5         C(B b){this.b=b;}
6         int foo(){return 42;}
7     }
8 }
9 public class Exercise{
10    public static void main(String [] arg){
11        assert ([??].foo()==42);
12    }
13 }
```

```
1 new A.C(null)
```

[Go back](#)

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax4[hard]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{
3     class C{
4         int foo(){return 42;}
5     }
6 }
7 public class Exercise{
8
9    public static void main(String [] arg){
10        assert (new A() {
11            public int foo(){return 42;}
12        }.foo()==42);
13    }
14 }
```

```
1 new A() {
2     public int foo(){return 42;}
3 }
```

[Go back](#)

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection Instance1[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 class A{ int m(){return 1;} }
3 public class Exercise{
4    public static void main(String [] arg){
5        final A a=[??];
6        assert a.m()==1;
7    }
8 }
```

[Go back](#)

INTRODUCTION

SWEN221 2021 T1

Question: Introduction biggerEveryTime[Medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3
4     public static int foo=1;
5
6     public static void main(String [] arg){
7         int foo=10;
8         assert (10==[???]):"assertion10";
9         assert (11==[???]):"assertion11";
10        assert (12==[???]):"assertion12";
11        assert (13==[???]):"assertion13";
12    }
13 }
14 }
```



[Go back](#)

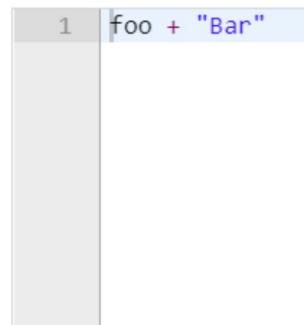
S

SWEN221 2021 T1

Question: Introduction ChangingTheStart[Medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3
4     public static int foo=1;
5
6     public static void main(String [] arg){
7         String foo="#";
8         assert ("#Bar".equals([???])):"assertion#Bar";
9         foo="_";
10        assert ("_Bar".equals([???])):"assertion_Bar";
11        foo="Bar";
12        assert ("BarBar".equals([???])):"assertionBarBar";
13    }
14 }
15 }
```



[Go back](#)

S

SWEN221 2021 T1

Question: Introduction DoubleEveryTime[Medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3
4     public static int foo=1;
5
6     public static void main(String [] arg){
7         String foo="#";
8         assert ("##".equals([???])):"assertion2";
9         assert ("####".equals([???])):"assertion4";
10        assert ("#####".equals([???])):"assertion8";
11    }
12 }
13 }
```

```
1 foo + "Bar"
```

Go back

S

SWEN221 2021 T1

Question: Introduction Scoping[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3     public static int x=10;
4     public static void main(String [] arg){
5         assert (11==[???]):"11 expected but"+[???];
6         int x=30;
7         assert (31==[???]):"31 expected but"+[???];
8     }
9 }
10 }
```

```
1 k+1
```

Go back

S

SWEN221 2021 T1

Question: Introduction SearchOnline[medium]

Please answer the following question:

```
1 //The answer must have balanced parentheses and not use "Float" or "Doub
2 public class Exercise{
3
4     public static void main(String [] arg){
5         float f=[???];
6         assert (Float.MAX_VALUE==f);
7     }
8 }
9
```

[Go back](#)

S

1 0x1.fffffeP+127f

SWEN221 2021 T1

Question: Introduction Simple[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3     public static void main(String [] arg){
4         double n=(([???])/2)*2;
5         assert ([??])== n;
6     }
7 }
8
```

[Go back](#)

S

1 10

SWEN221 2021 T1

Question: Introduction Simple?[hard]

Please answer the following question:

1 5

```
1 //The answer must have balanced parentheses
2 public class Exercise{
3     public static void main(String [] arg){
4         double n=(([???])/2)*2;
5         assert ([???])!= n;
6     }
7 }
8
```

Go back

s

SWEN221 2021 T1

Question: Introduction +Infinity[medium]

1.0f / 0.0f

Please answer the following question:

```
1 //The answer must have balanced parentheses and not use "Float" or "Doub
2 public class Exercise{
3
4     public static void main(String [] arg){
5         assert (Float.POSITIVE_INFINITY==[???]);
6     }
7 }
8
```

Go back

s

SWEN221 2021 T1

Question: Introduction -Infinity[medium]

Please answer the following question:

1 -1.0f / 0.0f

```
1 //The answer must have balanced parentheses and not use "Float" or "Doub"
2 public class Exercise{
3
4     public static void main(String [] arg){
5         assert (Float.NEGATIVE_INFINITY==[???]);
6     }
7 }
```

Go back

S

SWEN221 2021 T1

Question: Introduction FloatsAreSurprising[hard]

Please answer the following question:

1 0.0f / 0.0f

```
1 //The answer must have balanced parentheses and not use "Float" or "Doub"
2 public class Exercise{
3
4     public static void main(String [] arg){
5         float f=[???];
6         assert (f!=f);
7     }
8 }
```

Go back

S

SWEN221 2021 T1

Question: Introduction MyMax[easy]

Please answer the following question:

```
1 //The answer must have balanced parentheses, and not containing "exercis
2 import java.util.*;
3 public class Exercise{
4     public static float myMax(List<Float> elems){[???]}
5     //compute the max, undefined on empty lists
6     public static float exerciseMax(List<Float> elems) {
7         /*omitted*/
8     }
9     public static void main(String [] arg){
10        List<List<Float>> tests=Arrays.asList(
11            Arrays.asList(1f,2f,3f,4f,5f,6f,7f),
12            Arrays.asList(-1f,-2f,-3f,-7f,-13f),
13            Arrays.asList(-0.5f,0f,0f,30f,-100f,200f,23f,42f),
14            Arrays.asList(-12f/3f,32f,12f,45f,-1000f,99f),
15            Arrays.asList(13f/0f,-14f/1f,-200f/0f,300f,9001f),
16            Arrays.asList(-13f/0f,-100f/0f),
17            Arrays.asList(7f,6f,4f,3f,2f,1f,0f,1f,2f,3f,4f,5f,6f,7f,8f)
18        );
19        for(List<Float> ls:tests)
20            assert myMax(ls)==exerciseMax(ls): "error on "+ls;
21    }
22 }
```

```
1 float max = elems.get(0);
2 for (int i = 0; i < elems.size(); ++i) {
3     if (elems.get(i) > max) {
4         max = elems.get(i);
5     }
6 }
7 return max;
```