CONTRACTS

Question: Equality / Contracts BeDifferent[hard]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{
  public static void main(String [] arg){
    String s1=("hi" + null);
    String s2=("hi" + null);
    assert s1!=s2;
  }
}
```

```
"hi" + null
```

SWEN221 2021 T1

Question: Equality / Contracts Boolean1[easy]

Please answer the following question:

```
//The answer must be either true or false
class Point { int x; int y;
  public Point(int x, int y) {this.x = x; this.y = y;}
}
public class Exercise{

  public static void main(String [] arg){
    Point p1 = new Point(1,2);
    Point p2 = p1;
    p2.x = 2;
    if([???]){ assert p1==p2;}
    else {assert p1!=p2;}
  }
}
```

```
p1.x == 2
```

SWEN221 2021 T1

Question: Equality / Contracts Boolean2[easy]

Please answer the following question:

```
//The answer must be either true or false
class Point { int x; int y;
  public Point(int x, int y) {this.x = x; this.y = y;}
}
public class Exercise{

  public static void main(String [] arg){
    Point p1 = new Point(1,2);
    Point p2 = new Point(1,2);
    p2.x = 2;
    if([???]){ assert p1==p2;}
    else {assert p1!=p2;}
  }
}
```

Go back

```
p1.x == 2
```

SWEN221 2021 T1

Question: Equality / Contracts Boolean3[easy]

Please answer the following question:

```
//The answer must be either true or false
class Point { int x; int y;
  public Point(int x, int y) {this.x = x; this.y = y;}
}
public class Exercise{

  public static void main(String [] arg){
    Point p1 = new Point(1,2);
    Point p2 = new Point(1,2);
    if([???]){ assert p1==p2;}
    else {assert p1!=p2;}
  }
}
```

```
p1 == p2
```

Go back

SWEN221 2021 T1

Question: Equality / Contracts Boolean4[easy]

Please answer the following question:

```
//The answer must be either true or false
class Point { int x; int y;
  public Point(int x, int y) {this.x = x; this.y = y;}
}
public class Exercise{

  public static void main(String [] arg){
    Point p1 = new Point(1,2);
    Point p2 = new Point(1,2);
    if([???]){ assert p1.equals(p2);}
    else {assert !p1.equals(p2);}
  }
}
```

```
p1.equals(p2)
```

SWEN221 2021 T1

Question: Equality / Contracts Equals1[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class Point { int x; int y;
  public Point(int x, int y) {this.x = x; this.y = y;}
  [???]
}
public class Exercise{

  public static void main(String [] arg){
    Point p1 = new Point(1,2);
    Point p2 = new Point(1,2);
    assert p1.equals(p2);
  }
}
```

Go back

```
public boolean equals( Object obj )
{
   if ( obj instanceof Point )
   {
      Point point = (Point) obj ;
      return point.x == x && point.y == y ;
   }
   else
   {
      return false ;
   }
}
```

SWEN221 2021 T1

Question: Equality / Contracts Equals3[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class Point { int x; int y;
  public Point(int x, int y) {this.x = x; this.y = y;}
  [???]
}
public class Exercise{

  public static void main(String [] arg){
    Point p1 = new Point(1,2);
    Point p2 = new Point(1,2);
    assert p1.equals(p2);
    p2.x=10;
    assert !p1.equals(p2);
    assert !p1.equals(new Object());
    assert !p1.equals(new String());
    assert !p1.equals(Point.class);
  }
}
```

Go back

```
public boolean equals( Object obj )
{
   if ( obj instanceof Point )
   {
      Point point = (Point) obj ;
      return point.x == x && point.y == y ;
   }
   else
   {
      return false ;
   }
}
```

ddd

SWEN221 2021 T1

Question: Equality / Contracts Equals3[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class Point { int x; int y;
  public Point(int x, int y) {this.x = x; this.y = y;}
  [???]
}
public class Exercise{

  public static void main(String [] arg){
    Point p1 = new Point(1,2);
    Point p2 = new Point(1,2);
    assert p1.equals(p2);
    p2.x=10;
    assert !p1.equals(p2);
    assert !p1.equals(new Object());
    assert !p1.equals(new String());
    assert !p1.equals(Point.class);
  }
}
```

Go back

```
public boolean equals( Object obj )
{
   if ( obj instanceof Point )
   {
      Point point = (Point) obj ;
      return point.x == x && point.y == y ;
   }
   else
   {
      return false ;
   }
}
```

TIME_1

```
//The answer must have balanced parentheses
//Complete class Time24 adding add1Hour()
class Time24 {
  //toString,equals and hashCode as generated by eclipse
/*omitted*/
  final int seconds;
  final int minutes;
  final int hours;
  Time24(int seconds, int minutes, int hours){
    this.seconds=limit60(seconds);
    this.minutes = limit60(minutes);
    this.hours=limitHours(hours);
  }
  private static int limit60(int v){// 0..59 ok
    if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "
    return v;
  }
  private static int limitHours(int v){//0..23 ok
    if(v<0|| v>23){throw new IllegalArgumentException("limitHours ove
    return v;
  }
  static final Time24 twelve= new Time24(0,0,12);
  [???]
}

public class Exercise{
  public static void checkAdd(Time24 t){
    Time24 test=t;
    for(int i=0;i<24;i++){test=test.add1Hour();}
    assert test.equals(t):
    "if I add 24 hours, is the same as before for:"+t;
  }
  public static void main(String[]arg){
    checkAdd(Time24.twelve);
    checkAdd(new Time24(10,10,10));
    checkAdd(new Time24(59,59,23));
  }
}
```

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

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

TIME_2

```
//Complete class Time24 adding add1Hour(), remove1Hour()
class Time24 {
  //toString,equals and hashCode as generated by eclipse
/*omitted*/
  final int seconds;
  final int minutes;
  final int hours;
  Time24(int seconds, int minutes, int hours){
    this.seconds=limit60(seconds);
    this.minutes = limit60(minutes);
    this.hours=limitHours(hours);
  }
  private static int limit60(int v){// 0..59 ok
    if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "
    return v;
  }
  private static int limitHours(int v){//0..23 ok
    if(v<0|| v>23){throw new IllegalArgumentException("limitHours ove
    return v;
  }
  static final Time24 twelve= new Time24(0,0,12);
  [???]
}

public class Exercise{
  public static void checkAddRemove(Time24 t,int howMuch){
    Time24 test=t;
    for(int i=0;i<howMuch;i++){test=test.add1Hour();}
    for(int i=0;i<howMuch;i++){test=test.remove1Hour();}
    assert test.equals(t):
    "for hours: if I add "+howMuch+" and I remove "+howMuch+" is the
    }
  public static void checkAdd(Time24 t){
    {Time24 test=t;
    for(int i=0;i<24;i++){test=test.add1Hour();}
    assert test.equals(t):
    "if I add 24 hours, is the same as before for:"+t;}
  }
  public static void checkRemove(Time24 t){//as for checkAdd
    /*omitted*/
  }
  public static void checkAll(Time24 t){
    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

```
//The answer must have balanced parentheses
//HINT: Reuse the code of the former question!
//Complete class Time24 adding add1Hour(), remove1Hour()
//add1Minute(), remove1Minute()
class Time24 {
  //toString,equals and hashCode as generated by eclipse
/*omitted*/
  final int seconds;
  final int minutes;
  final int hours;
  Time24(int seconds, int minutes, int hours){
    this.seconds=limit60(seconds);
    this.minutes = limit60(minutes);
    this.hours=limitHours(hours);
  }
  private static int limit60(int v){// 0..59 ok
    if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "+v)
    return v;
  }
  private static int limitHours(int v){//0..23 ok
    if(v<0|| v>23){throw new IllegalArgumentException("limitHours over "
    return v;
  }
  static final Time24 twelve= new Time24(0,0,12);
  [???]
}

public class Exercise{
  /*omitted*/
  public static void main(String[]arg){
    /*omitted*/
  }
}
```

```
public Time24 add1Hour() {
    int updatedHour = hours + 1;
    if (updatedHour >= 24) {
        updatedHour %= 24;
    }
    return new Time24(seconds, minutes, updatedHour);
}

public Time24 remove1Hour() {
    int updatedHour = hours - 1;
    if (updatedHour < 0) {
        updatedHour += 24;
    }
    return new Time24(seconds, minutes, updatedHour);
}

public Time24 add1Minute() {
    int updatedMinutes = minutes + 1;
    int updatedHour = hours;
    if (updatedMinutes >= 60) {
        updatedMinutes %= 60;
        updatedHour = hours + 1;
        if (updatedHour >= 24) {
            updatedHour %= 24;
        }
    }
    return new Time24(seconds, updatedMinutes, updatedHour);
}

public Time24 remove1Minute() {
    int updatedMinutes = minutes - 1;
    int updatedHour = hours;
    if (updatedMinutes < 0) {
        updatedMinutes += 60;
        updatedHour = hours - 1;
        if (updatedHour < 0) {
            updatedHour += 24;
        }
    }
    return new Time24(seconds, updatedMinutes, updatedHour);
}
```

TIME_4

```java
//The answer must have balanced parentheses
//HINT: Reuse the code of the former question!
//Complete class Time24 adding add1Hour(), remove1Hour()
//add1Minute(), remove1Minute(), add1Second(), remove1Second()
class Time24 {
  //toString,equals and hashCode as generated by eclipse
/*omitted*/
  final int seconds;
  final int minutes;
  final int hours;
  Time24(int seconds, int minutes, int hours){
    this.seconds=limit60(seconds);
    this.minutes = limit60(minutes);
    this.hours=limitHours(hours);
  }
  private static int limit60(int v){// 0..59 ok
    if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "+v)
    return v;
  }
  private static int limitHours(int v){//0..23 ok
    if(v<0|| v>23){throw new IllegalArgumentException("limitHours over "
    return v;
  }
  static final Time24 twelve= new Time24(0,0,12);
  [???]
}

public class Exercise{
  /*omitted*/
  public static void main(String[]arg){
    /*omitted*/
  }
```

TIME_5

```
//The answer must have balanced parentheses
//HINT: Reuse the code of the former question!
//Complete class Time12 adding add1Hour(), remove1Hour()
//add1Minute(), remove1Minute(), add1Second(), remove1Second()
class Time12 {
  //toString,equals and hashCode as generated by eclipse
  /*omitted*/
  final int seconds;
  final int minutes;
  final int hours;
  final boolean isPm;
  Time12(int seconds, int minutes, int hours,boolean isPm){
    this.seconds=limit60(seconds);
    this.minutes = limit60(minutes);
    this.hours=limitHours(hours);
    this.isPm=isPm;
  }
  private static int limit60(int v){// 0..59 ok
    if(v<0|| v>59){throw new IllegalArgumentException("limit60 over "+v)
    return v;}
  private static int limitHours(int v){//1..12 ok
    if(v<1|| v>12){throw new IllegalArgumentException("limitHours over "
    return v;}
  static final Time12 twelve= new Time12(0,0,12,true);
  [???]
  }
  public class Exercise{
    /*omitted*/
  }
  public static void main(String[]arg){
    /*omitted*/
  }
}
```

s

EXCEPTIONS:

SWEN221 2021 T1

Question: Exceptions AfterTry[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class EmptyArrayException extends Exception{}
class ArrayUtil{
  public static Object lastElement (Object[]array)throws EmptyArrayExcep
    if(array.length>0)return array[array.length-1];
    throw new EmptyArrayException();
  }
}
public class Exercise{
  public static void test(){
    try{
      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.*/}
  }
}
```

Go back

Enter your code here:

```
catch (Exception e) {

}
```

SEND CLEAR

S

SWEN221 2021 T1

Question: Exceptions Declaring1[easy]

Please answer the following question:

```
//The answer must have no { and }
class EmptyArrayException extends [???]{}
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 main(String [] arg){
    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);
  }
}
```

Go back

Enter your code here:

```
RuntimeException
```

SEND CLEAR

S

SWEN221 2021 T1

Question: Exceptions Declaring3[medium]

Please answer the following question:

```
//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.*/}
  }
}
```

Go back

Enter your code here:

```
throws EmptyArrayException
```

SEND CLEAR

S

SWEN221 2021 T1

Question: Exceptions GoodToKnow[hard]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{
  public static int a()throws Throwable{
    return 1;
  }
  public static int b(){
    try{ return b();}
    catch(Throwable t){throw new Error(t);}
    [???]
  }
  public static void main(String [] arg){
    assert b()==0;
  }
}
```

Enter your code here:

```
finally {
    return 0;
}
```

SEND CLEAR

S

Question: Exceptions MakeADifference[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{

  public static int m1(){
    try{
      try{if(true){throw [???]}}
      catch(Error e){return 2;}
      finally{return 2;}
      }
    catch(Throwable t){return 10;}
   }
  public static int m2(){
    try{
      try{if(true){throw [???]}}
      catch(Error e){return 2;}
      return 2;
      }
    catch(Throwable t){return 10;}
   }
  public static void main(String [] arg){
    assert (m1()!=m2()):"assertion:"+m1()+"!="+m2();
  }
}

```

Enter your code here:

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

S

Question: Exceptions Person1[medium]

Please answer the following question:

```

//The answer must have balanced parentheses
//Declare a class Person with fields String name and int age.
//Declare a constructor initializing the fields and
//Override the method toString to produce the reqired results.
import java.util.HashMap;
import java.util.List;
import java.util.ArrayList;
import java.util.Arrays;
class Person {
    String name ;
    int age ;

    public Person(String name, int age) {
        super();
        this.name = name;
        this.age = age;

    }

    @Override
    public String toString() {
        return name + ":" + age ;
    }
}
public class Exercise{
  public static void main(String [] arg){
    assert "Marco:33".equals(new Person("Marco",33).toString());
    assert "Adam:13".equals(new Person("Adam",13).toString());
    assert "Godzilla:450000000".equals(new Person("Godzilla",450000000).
    assert "Lidia:83".equals(new Person("Lidia",83).toString());
  }
}

```

Enter your code here:

```
class Person {
    String name ;
    int age ;

    public Person(String name, int age) {
        super();
        this.name = name;
        this.age = age;

    }

    @Override
    public String toString() {
        return name + ":" + age ;
    }
}
```

SEND CLEAR

S

Question: Exceptions Person2[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
//As for the former question, declare a class Person with fields String
//Declare a constructor initializing the fields and
//Override the method toString to produce the reqired results.
//In addition: check as an invariant that the age can not
//be negative, and the name can not be the empty string.
//In case this invariant is violated, throw IllegalArgumentException
import java.util.HashMap;
import java.util.List;
import java.util.ArrayList;
import java.util.Arrays;
[???]
public class Exercise{
  public static void main(String [] arg){
    assert "Marco:33".equals(new Person("Marco",33).toString());
    assert "Adam:13".equals(new Person("Adam",13).toString());
    assert "Godzilla:450000000".equals(new Person("Godzilla",450000000).
    assert "Lidia:83".equals(new Person("Lidia",83).toString());
    try{new Person("Lidia",-1);assert false;}catch(IllegalArgumentExcept
    try{new Person("Adam",-100);assert false;}catch(IllegalArgumentExce
    try{new Person("",100);assert false;}catch(IllegalArgumentException
    try{new Person("",10);assert false;}catch(IllegalArgumentException e
    try{new Person("",-100);assert false;}catch(IllegalArgumentExceptio
  }
}
```

```
class Person {
   String name;
   int age;

   public Person(String n,int a){
       setName(n);
       setAge(a);
   }

   public void setName(String name) throws IllegalArgumentException
   {
       if(name=="")
           throw new IllegalArgumentException("Name cannot be an empty s
       else
           this.name=name;
   }

   public void setAge(int age) throws IllegalArgumentException
   {
       if(age<0)
           throw new IllegalArgumentException("Age cannot be negative");
       else
           this.age=age;
   }

   @Override
   public String toString() {
       return name + ":" + age;
   }
}
```

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:

```
// The answer must have balanced parentheses
class Box<[???]> {
  private [???] contents;

  public Box([???] contents) {
    this.contents = contents;
  }

  public [???] getContents() {
    return contents;
  }
}

public class Exercise {
  public static void main(String[] args) {
    Box<Integer> intBox = new Box<Integer>(1);
    assert intBox.getContents().equals(1);
    Box<String> stringBox = new Box<String>("Dave");
    assert stringBox.getContents().equals("Dave");
  }
}

```

Go back

Enter your code here:

```
T
```

SEND CLEAR

S

Question: Generics Generic2[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
class NumberBox<T [???] Number> {
  private T contents;

  public NumberBox(T contents) {
    this.contents = contents;
  }

  public T getContents() {
    return contents;
  }
}

public class Exercise {
  public static void main(String[] args) {
    NumberBox<Integer> intBox = new NumberBox<Integer>(1);
    assert intBox.getContents().equals(1);
    NumberBox<Float> floatBox = new NumberBox<Float>(3.145f);
    assert floatBox.getContents().equals(3.145f);
  }
}

```

Enter your code here:

```
extends
```

SEND CLEAR

S

Question: Generics Generic3[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
class Pair [???] {
  private S first;
  private T second;

  public Pair(S first, T second) {
    this.first = first;
    this.second = second;
  }

  public S getFirst() { return first; }

  public T getSecond() { return second; }
}

public class Exercise {
  public static void main(String[] args) {
    Pair<String,Integer> p1 = new Pair<String,Integer>("Hello",1);
    assert p1.getFirst().equals("Hello");
    assert p1.getSecond().equals(1);
  }
}
```

Enter your code here:

```
<S, T>
```

S

Question: Generics Generic4[hard]

Please answer the following question:

```
// The answer must have balanced parentheses
import java.util.*;
public class Exercise {

  static String get(List<? [???]> strings, int element) {
    return strings.get(element);
  }

  public static void main(String[] args) {
    ArrayList<String> strings = new ArrayList<String>();
    strings.add("Hello");
    strings.add("Dave");
    assert get(strings,0).equals("Hello");
    assert get(strings,1).equals("Dave");
  }
}
```

Enter your code here:

```
extends String
```

S

Question: Generics Generic5[hard]

Please answer the following question:

```
// The answer must have balanced parentheses
import java.util.*;
public class Exercise {

  static [???] String get(List<T> strings, int element) {
    return strings.get(element);
  }

  public static void main(String[] args) {
    ArrayList<String> strings = new ArrayList<String>();
    strings.add("Hello");
    strings.add("Dave");
    assert get(strings,0).equals("Hello");
    assert get(strings,1).equals("Dave");
  }
}
```

```
<T extends String>
```

S

Question: Generics Generic6[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
import java.util.*;

interface Father {
  public boolean isFather();
}

interface Mother {
  public boolean isMother();
}

class Child implements Mother,Father {
  public boolean isMother() { return false; }
  public boolean isFather() { return false; }
}

public class Exercise {
  @SuppressWarnings("unchecked")
  static <T extends Mother [???]> T f() {
    return (T) new Child();
  }

  public static void main(String[] args) {
    assert f().isMother() == false;
    assert f().isFather() == false;
  }
}
```

```
& Father
```

S

```java
// The answer must have balanced parentheses
import java.util.ArrayList;
import java.util.List;

interface Animal { }

class Cat implements Animal {
  public String toString() {
    return "Cat";
  }
}

class Dog implements Animal {
  public String toString() {
    return "Dog";
  }
}

public class Exercise {

  public static List<[???]> createList(String... names) {
    ArrayList<[???]> r = new ArrayList<[???]>();
    for(String name : names) {
      switch(name) {
        case "Dog":
          r.add(new Dog());
          break;
        case "Cat":
          r.add(new Cat());
          break;
      }
    }
    return r;
  }

  public static void main(String[] args) {
    assert createList("Cat").toString().equals("[Cat]");
    assert createList("Dog").toString().equals("[Dog]");
    assert createList("Cat","Dog").toString().equals("[Cat, Dog]");
  }
}
```

```
Animal
```

S

```java
// The answer must have balanced parentheses
import java.util.*;

class Parent {
  public List<? [???] Child> get() {
    ArrayList<Parent> r = new ArrayList<Parent>();
    r.add(this);
    return r;
  }
  public String toString() { return "Parent"; }
}

class Child extends Parent {
  public List<? [???] Parent> get() {
    ArrayList<Object> r = new ArrayList<Object>();
    r.add(this);
    return r;
  }
  public String toString() { return "Child"; }
}

public class Exercise {

  public static void main(String[] args) {
    Parent parent = new Parent();
    Child child = new Child();
    assert parent.get().toString().equals("[Parent]");
    assert child.get().toString().equals("[Child]");
  }
}
```

```
super
```

S

Question: Generics ApplicableMethod[hard]

Please answer the following question:

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

```
(List<? extends Object> sl) {
    StringBuilder sb = new StringBuilder();

    for(Object a:sl){
        sb.append(a);
    }
    return sb.toString();
}
```

S

Question: Generics Maps1[medium]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "+"
//Write a function that takes a map CityName -> Population
//and return the name of the most populated city
import java.util.HashMap;
import java.util.List;
import java.util.Arrays;
public class Exercise{
  public static  String findMostPopulated(HashMap<String,Integer> that){
    [???]
  }
  public static void check(String expected,List<String>city,List<Integer
    HashMap<String,Integer>map=new HashMap<>();
    for(int i=0;i<city.size();i++){ map.put(city.get(i),population.get(i
    assert expected.equals(findMostPopulated(map)):"expected "+expected+
  }
  public static void main(String [] arg){
    check("Rome",Arrays.asList("Rome","Milan","Venice"),
      Arrays.asList(500,100,10));
    check("Milan",Arrays.asList("Rome","Milan","Venice"),
      Arrays.asList(100,500,10));
    check("Milan",Arrays.asList("Milan"),
      Arrays.asList(100));
    try{findMostPopulated(new HashMap<String,Integer>());assert false;}c
  }
}
```

```
int mostPop = 0;
    String mostPopCity = "";
    if(that.isEmpty()){
        throw new java.util.NoSuchElementException();
    }
    for(String key : that.keySet() ){
        if(that.get(key) >= mostPop){
            mostPop = that.get(key);
            mostPopCity = key;
        }
    }
    return mostPopCity;
```

S

```
//The answer must have balanced parentheses
class Point{ int x;int y;
  Point(int x, int y){this.x=x;this.y=y;}
}
class ColPoint extends Point{ int colour;
  ColPoint(int x, int y,int colour){
    super(x,y);this.colour=colour;}
}
public class Exercise{
  public static void main(String [] arg){
    ColPoint c1 = new ColPoint(1,2,124);
    ColPoint c2 = new ColPoint(1,3,120);
    Point c3 = new Point(1,5);
    assert(c1 == min(c1,c3));
    c2 = min(c1,c2);
    assert(c1 == c2);
  }
[???]
}
```

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

S

```java
//The answer must have balanced parentheses
//Extend class A<T> so that findMax finds
//the smallest Integer number in the list
import java.util.Arrays;
import java.util.List;
abstract class A<T>{
  T findMax(List<T> that){
    if(that.isEmpty()){throw new java.util.NoSuchElementException();}
    T candidate=that.get(0);
    for(T s:that){
      candidate=better(candidate,s);
    }
    return candidate;
  }
abstract T better(T e1, T e2);
}

class B extends A<Integer>{
  Integer better(Integer e1, Integer e2){
    return Math.min(e1, e2);
  }
}

public class Exercise{
  static void check(Integer expected,List<Integer> data){
  Integer result=new B().findMax(data);
  assert expected.equals(result): "on data: "+data+"expected: "+expected
  }
  public static void main(String [] arg){
    check(3,Arrays.asList(22,12,13,14,55,102,13,3));
    check(-12,Arrays.asList(22,-12,13,14,55,102,13,3));
    check(1,Arrays.asList(1));
  }
}
```

```java
return Math.min(e1, e2);
```

S

```java
//Extend class A<T> so that findMax finds
//the tallest student in the list
import java.util.Arrays;
import java.util.List;
abstract class A<T>{
  T findMax(List<T> that){
    if(that.isEmpty()){throw new java.util.NoSuchElem
    T candidate=that.get(0);
    for(T s:that){
      candidate=better(candidate,s);
    }
    return candidate;
  }
  abstract T better(T e1, T e2);
}

class Student{
  final String name;
  final float averageMark;
  final int height;
  Student(String name, float averageMark, int height)
    this.name=name;
    this.averageMark=averageMark;
    this.height=height;
  }
  public String toString(){
    return "S("+name+","+averageMark+","+height+")";
  }
}

class B extends A<Student>{
  [???]
}

public class Exercise{
  static void check(String expected, Student ... data
    Student result=new B().findMax(Arrays.asList(data
    assert expected.equals(result.name):
      "on data: "+data+"expected: "+expected+" result
  }
  public static void main(String [] arg){
    check("Paul",
      new Student("Paul",10,185),
```

```java
Student better(Student e1, Student e2) {
    if(e1.height<=e2.height){
        return e2;
    } else {
        return e1;
    }
}
```

S

```
//The answer must have balanced parentheses
//Extend class A<T> so that findMax finds
//the student whose name is last in alphabetical order
import java.util.Arrays;
import java.util.List;
abstract class A<T>{
  T findMax(List<T> that){
    if(that.isEmpty()){throw new java.util.NoSuchElementException();}
    T candidate=that.get(0);
    for(T s:that){
      candidate=better(candidate,s);
    }
    return candidate;
  }
abstract T better(T e1, T e2);
}

class Student{
  final String name; final float averageMark; final int height;
  Student(String name, float averageMark, int height){
    this.name=name;this.averageMark=averageMark; this.height=height;}
  public String toString(){return "S("+name+","+averageMark+","+height+"
  }

class B extends A<Student>{
  [???]
}

public class Exercise{
  static Student S(String name, float averageMark, int height){
    return new Student(name,averageMark,height);}
  static void check(String expected,List<Student> data){
    Student result=new B().findMax(data);
    assert expected.equals(result.name): "on data: "+data+"expected: "+e
  }
  public static void main(String [] arg){
    check("Zorro", Arrays.asList(S("Adam",10,185),S("Zorro",8,145)));
    check("Paul", Arrays.asList(S("Paul",10,185),S("Lan",8,145),S("Marc
    check("Thor", Arrays.asList(S("Paul",10,185),S("Thor",10,215),S("Mar
    check("Godzilla", Arrays.asList(S("Godzilla",1,6500)));
  }
}
```

```
Student better(Student e1, Student e2) {
    if(e1.name.compareTo(e2.name)>0){
        return e1;
    } else if(e1.name.compareTo(e2.name)<=0){
        return e2;
    } else {
        return e1;
    }
}
```

S

```
//The answer must have balanced parentheses
//Extend class A<T> so that findMax finds
//the student with the highest mark
import java.util.Arrays;
import java.util.List;
abstract class A<T>{
  T findMax(List<T> that){
    if(that.isEmpty()){throw new java.util.NoSuchElementException();}
    T candidate=that.get(0);
    for(T s:that){
      candidate=better(candidate,s);
    }
    return candidate;
  }
abstract T better(T e1, T e2);
}

class Student{
  final String name; final float averageMark; final int height;
  Student(String name, float averageMark, int height){
    this.name=name;this.averageMark=averageMark; this.height=height;}
  public String toString(){return "S("+name+","+averageMark+","+height+"
}
[???]

public class Exercise{
  static Student S(String name, float averageMark, int height){
    return new Student(name,averageMark,height);}
  static void check(String expected,List<Student> data){
    Student result=new B().findMax(data);
    assert expected.equals(result.name): "on data: "+data+"expected: "+e
  }
  public static void main(String [] arg){
    check("Adam", Arrays.asList(S("Adam",10,185),S("Zorro",8,145)));
    check("Paul", Arrays.asList(S("Paul",10,185),S("Lan",8,145),S("Marc
    check("Paul", Arrays.asList(S("Paul",10,185),S("Thor",1,215),S("Mar
    check("Godzilla", Arrays.asList(S("Godzilla",1,6500)));
  }
}
```

```
class B extends A<Student>{
    Student better(Student e1, Student e2) {
        if (e1.averageMark>=e2.averageMark){
            return e1;
        } else {
            return e2;
        }
    }
}
```

S

```
//The answer must have balanced parentheses
//Extend class A<T> so that findMax finds
//the person having the shortest address
//(you have also to define the person class!)
import java.util.Arrays;
import java.util.List;
abstract class A<T>{
  T findMax(List<T> that){
    if(that.isEmpty()){throw new java.util.NoSuchElementException();}
    T candidate=that.get(0);
    for(T s:that){
      candidate=better(candidate,s);
    }
    return candidate;
  }
abstract T better(T e1, T e2);
}
[???]

public class Exercise{
  static Person P(String name, String address){
    return new Person(name,address);}
  static void check(String expected,List<Person> data){
    Person result=new B().findMax(data);
    assert expected.equals(result.name): "on data: "+data+"expected: "+e
  }
  public static void main(String [] arg){
    check("Zorro", Arrays.asList(P("Adam","saint monique road 234"),P("z
    check("Lan", Arrays.asList(P("Paul","google road 245 california"),P(
    check("Godzilla", Arrays.asList(P("Godzilla","under the sea")));
  }
}
```

```
class Person {
    String name,address;
    public Person(String name, String address) {
        this.name = name;
        this.address = address;
    }
}

class B extends A<Person>{

    @Override
    Person better(Person e1, Person e2) {
        if(e1.address.length()<=e2.address.length()){
            return e1;
        } else {
            return e2;
        }
    }
}
```

S

```
//The answer must have balanced parentheses
//Extend class A<T> so that findMax finds
//the student whose mark is the nearest to
//a given target
import java.util.Arrays;
import java.util.List;
abstract class A<T>{
  T findMax(List<T> that){
    if(that.isEmpty()){throw new java.util.NoSuchElementException();}
    T candidate=that.get(0);
    for(T s:that){
      candidate=better(candidate,s);
    }
    return candidate;
  }
abstract T better(T e1, T e2);
}

class Student{
  final String name; final float averageMark; final int height;
  Student(String name, float averageMark, int height){
    this.name=name;this.averageMark=averageMark; this.height=height;}
  public String toString(){return "S("+name+","+averageMark+","+height+"
  }
  [???]

public class Exercise{
  static Student S(String name, float averageMark, int height){
    return new Student(name,averageMark,height);}
  static void check(int target,String expected,List<Student> data){
    Student result=new B(target).findMax(data);
    assert expected.equals(result.name): "on data: "+data+"expected: "+e
  }
  public static void main(String [] arg){
    check(5,"Zorro", Arrays.asList(S("Adam",10,185),S("Zorro",8,145)));
    check(3,"Lan", Arrays.asList(S("Paul",10,185),S("Lan",2,145),S("Marc
    check(7,"Thor", Arrays.asList(S("Paul",10,185),S("Thor",7,215),S("Ma
    check(2,"Godzilla", Arrays.asList(S("Godzilla",1,6500)));
  }
}
```

```
class B extends A<Student>{
    int target;

    public B(int target) {
        this.target = target;
    }

    @Override
    Student better(Student e1, Student e2) {
        if(e1.averageMark-target<=e2.averageMark-target){
            return e1;
        } else {
            return e2;
        }
    }
}
```

S

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 }
12
```

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 }
12
```

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:

```
//The answer must have balanced parentheses
class A{
  private int f=0;
  public void setF(int f){[???]}
  public boolean checkF(int expected){return f==expected;}
}
class B extends A{
  public void m(){
    setF(3);
    assert checkF(3);
    setF(8);
    assert checkF(8);
  }
}
public class Exercise{
  public static void main(String [] arg){
    new B().m();
  }
}
```

Go back

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

SWEN221 2021 T1

Question: Inheritance Field2[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{
  private int f=0;
  public void setF(int _f){f=_f;}
  public boolean checkF(int expected){return f==expected;}
}

class B extends A{
  public void setF(int f){[???]}
  public void m(){
    setF(3);
    assert checkF(-3);
    setF(8);
    assert checkF(-8);
  }
}

public class Exercise{
  public static void main(String [] arg){
    new B().m();
  }
}
```

Go back

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

Question: Inheritance Interface1[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
interface Student {
  public String getName();
  public int getID();
}

class MyStudent [???] Student {
  private String name;
  private int id;

  public MyStudent(String name, int id) {
    this.name = name;
    this.id = id;
  }

  public String getName() { return name; }

  public int getID() { return id; }
}

public class Exercise {
  public static void main(String [] arg){
    Student s = new MyStudent("Dave",300384719);
    assert s.getName().equals("Dave");
    assert s.getID() == 300384719;
  }
}
```

Enter your code here:

```
implements
```

SEND CLEAR

Question: Inheritance Interface2[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
interface Person {
  public String getName();
}

interface Student [???] Person {
  public int getID();
}

class MyStudent implements Student {
  private String name;
  private int id;

  public MyStudent(String name, int id) {
    this.name = name;
    this.id = id;
  }

  public String getName() { return name; }

  public int getID() { return id; }
}

public class Exercise {
  public static void main(String [] arg){
    Student s = new MyStudent("Dave",300384719);
    assert s.getName().equals("Dave");
    assert s.getID() == 300384719;
  }
}
```

Enter your code here:

```
extends
```

Question: Inheritance Interface3[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
class Parent {
  public int f() { return 123; }
}

class Child extends Parent {
  public int f() {
    return [???].f();
  }
}

public class Exercise {
  public static void main(String[] args) {
    Child c = new Child();
    assert c.f() == 123;
  }
}
```

Enter your code here:

```
new Parent()
```

SWEN221 2021 T1

Question: Inheritance symbolic1[hard]

Please answer the following question:

```
//The answer must contains only digits
abstract class Parent {
  int x=100;
  Parent (int x) {this.x=x;}
  abstract int m(int x);
  static Parent k (Parent x) { return x;}
  int f (int x) { return m(x+1);}
}

class Heir extends Parent {
  static int x=200;
  Heir (int x) { super (x+1);}
  Heir () {this(3);}
  int m(int x) { return x+2;}
  int n (int x) { return this.x + x; }
}

public class Exercise{
  public static void main(String[] arg){
    Parent p = new Heir();
    Heir h = new Heir();
    assert(p.f(h.x)==[???]);
  }
}
```

Go back

```
203
```

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
```

Go back

```
1  9
```

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 }
25
```

Go back

```
1  4
```

S

MISCELLANEOUS

SWEN221 2021 T1

Question: Misc Concat[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
//write a function concat all the String in a list,
//adding "[" at the start and "]" at the end
import java.util.ArrayList;
import java.util.Arrays;
public class Exercise{
  public static  String concat(ArrayList<String> that){
    [???]
  }
  public static void main(String [] arg){
    assert "[abc]".equals(
      concat(new ArrayList<String>(Arrays.asList("","a","b","c"))));
    assert "[]".equals(concat(new ArrayList<String>()));
    /*omitted*/
  }
}
```

Go back

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

S

SWEN221 2021 T1

Question: Misc IntegerList[easy]

Please answer the following question:

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

Go back

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

S

SWEN221 2021 T1

Question: Misc LongestString[medium]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "+"
//write a function that returns longest string in a list,
import java.util.ArrayList;
import java.util.Arrays;
public class Exercise{
  public static  String longest(ArrayList<String> that){
    [???]
  }
  public static void main(String [] arg){
    assert "foo".equals(
      longest(new ArrayList<String>(Arrays.asList("","a","foo","b"))));
    assert "bar".equals(
      longest(new ArrayList<String>(Arrays.asList("","bar","b","b"))));
    /*omitted*/
  }
}
```

Go back

```
if (that.size() == 0) {
    throw new java.util.NoSuchElementException();
}

String longest = "";

int i = 0;
for (String s : that) {
    if (s.length() > i) {
        i = s.length();
        longest = s;
    }
}
return longest;
```

S

SWEN221 2021 T1

Question: Misc MyAverage[medium]

Please answer the following question:

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

Go back

```
float sum = 0;
for (int i = 0; i < elems.size(); ++i) {
    sum += elems.get(i);
}
return (sum / elems.size());
```

S

SWEN221 2021 T1

Question: Misc MyPow[medium]

Please answer the following question:

```
//The answer must have balanced parentheses, and not containing "exercis
//exercisePow is an implementation for the pow function
public class Exercise{
  public static int myPow(int base,int exp){[???]}
  public static int exercisePow(int base,int exp){
    /*omitted*/
    }
  public static void main(String [] arg){
    for(int i=0;i<15;i++)
      for(int j=0;j<5;j++)
        assert myPow(i,j)==exercisePow(i,j): "error on "+ i +"," + j + "
  }
}
```

Go back

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

S

SWEN221 2021 T1

Question: Misc SingleKeyword[medium]

Please answer the following question:

```
//The answer must be a single keyword
class A{
  [???] void foo(String s){assert false;}
  void foo(Object o){}
}
public class Exercise{
  public static void main(String [] arg){
    new A().foo("Hello");
  }
}
```

Go back

```
private
```

S

Question: Misc UseFindMax1[medium]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "+"
//Using the function findMax, remove the bigger element from the list
import java.util.ArrayList;
import java.util.List;
import java.util.Arrays;
public class Exercise{
  public static  Float findMax(List<Float> that){
    if(that.isEmpty()){throw new java.util.NoSuchElementException();}
    float candidate=that.get(0);
    for(Float f:that){
      if(f>candidate){candidate=f;}
    }
    return candidate;
  }
  public static  void removeMax(List<Float> that){
    [???]
  }
  public static void check(List<Float>original,List<Float>expected){
    original=new ArrayList<Float>(original);
    expected=new ArrayList<Float>(expected);
    removeMax(original);
    assert original.equals(expected);
  }
  public static void main(String [] arg){
    check(Arrays.asList(1f,2f,3f),Arrays.asList(1f,2f));
    check(Arrays.asList(100f),new ArrayList<Float>());
    check(Arrays.asList(4f,1f,2f,3f),Arrays.asList(1f,2f,3f));
    /*omitted*/
  }
}
```

Go back

```
Float max = findMax(that);
that.remove(max);
return longest;
```

S

Question: Misc UseFindMax2[medium]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "+"
//Using the function findMax, without modifying the orignal list,
//produce a new list that is like the original but do not have the bigge
import java.util.ArrayList;
import java.util.List;
import java.util.Arrays;
public class Exercise{
  public static  Float findMax(List<Float> that){
    if(that.isEmpty()){throw new java.util.NoSuchElementException();}
    float candidate=that.get(0);
    for(Float f:that){
      if(f>candidate){candidate=f;}
    }
    return candidate;
  }
  public static  List<Float> removeMax(List<Float> that){
    [???]
  }
  public static void check(List<Float>original,List<Float>expected){
    ArrayList<Float>original2=new ArrayList<Float>(original);
    expected=new ArrayList<Float>(expected);
    List<Float>result=removeMax(original);
    assert original.equals(original2);
    assert result.equals(expected);
  }
  public static void main(String [] arg){
    check(Arrays.asList(1f,2f,3f),Arrays.asList(1f,2f));
    check(Arrays.asList(100f),new ArrayList<Float>());
    check(Arrays.asList(4f,1f,2f,3f),Arrays.asList(1f,2f,3f));
    /*omitted*/
  }
}
```

```
Float max = findMax(that);
ArrayList<Float> newList = new ArrayList<>();
newList.addAll(that);
newList.remove(max);
return newList;
```

S

POLYMORPHISM

SWEN221 2021 T1

Question: Polymorphism ManyF[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{}
class B extends A{}
public class Exercise{
  public static boolean f(A p1, A p2){ return false;}
  public static boolean f(B p1, A p2){ return true;}
  public static boolean f(B p1, B p2){ return false;}
  public static void main(String [] arg){
    assert f([???]);
  }
}
```

Go back

```
new B(), new A()
```

S

SWEN221 2021 T1

Question: Polymorphism ManyM1[easy]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{}
class B extends A{}
public class Exercise{
  public static boolean m(A p1){ return false;}
  public static boolean m(String p1){ return true;}
  public static boolean m(B p1){ return false;}
  public static void main(String [] arg){
    assert m([???]);
  }
}
```

Go back

```
"Ricky"
```

S

SWEN221 2021 T1

Question: Polymorphism ManyM2[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{}
class B extends A{}
public class Exercise{
  public static boolean m(A p1, B p2){ return false;}
  public static boolean m(A p1, A p2){ return true;}
  public static boolean m(B p1, B p2){ return false;}
  public static void main(String [] arg){
    assert m([???]);
  }
}
```

Go back

```
new A(), new A()
```

S

SWEN221 2021 T1

Question: Polymorphism ManyMF[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{}
class B extends A{}
public class Exercise{
  public static boolean m(A p1, B p2){ return false;}
  public static boolean m(A p1, A p2){ return true;}
  public static boolean m(B p1, B p2){ return false;}
  public static boolean f(A p1, A p2){ return false;}
  public static boolean f(B p1, A p2){ return true;}
  public static boolean f(B p1, B p2){ return false;}
  public static void main(String [] arg){
    assert m([???]);
    assert f([???]);
  }
}
```

Go back

```
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 }
18
```

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 }
20
```

Go back

Photos - b.PNG

```
1  Parent()
```

S

SWEN221 2021 T1

Question: Polymorphism Poly3[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
interface Parent {
  [???]
}

class Child implements Parent {
  public String name(boolean b) {
    return "Child " + b;
  }

  public String name(int i) {
    return "Child " + i;
  }
}

public class Exercise {
  public static void main(String[] args) {
    Parent p = new Child();
    assert p.name(1).equals("Child 1");
    assert p.name(true).equals("Child true");
  }
}
```

Go back

```
public String name(boolean b);
public String name(int i);
```

S

SWEN221 2021 T1

Question: Polymorphism Poly4[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
interface Parent {
  public String name(Object b);
}

class Child implements Parent {
  public String name(Object b) {
    return [???];
  }
  public String name(String b) {
    return "Child " + [???];
  }
}

public class Exercise {
  public static void main(String[] args) {
    Parent p = new Child();
    assert p.name("String").equals("Child String");
  }
}
```

Go back

```
"Child String"
```

S

```java
// The answer must have balanced parentheses
// This is an example of "double dispatch"

interface Animal {
  public boolean eats(Animal o);
  public boolean isEatenBy(Animal o);
}

class Cat implements Animal {
  public boolean eats(Animal o) {
    return o.isEatenBy(this);
  }

  public boolean isEatenBy(Animal o) {
    [???]
  }
}

class Dog implements Animal {
  public boolean eats(Animal o) {
    return o.isEatenBy(this);
  }

  public boolean isEatenBy(Animal o) {
    [???]
  }
}

public class Exercise {
  public static void main(String[] args) {
    Animal a = new Dog();
    Animal b = new Cat();

    assert a.eats(b);
    assert !b.eats(b);
  }
}
```

```java
if (o == this) {
    return false;
}
return true;
```

S

Question: Polymorphism Poly6[medium]

Please answer the following question:

```
// The answer must have balanced parentheses

class Cat {
  public boolean isEatenBy(Object o) {
    return false;
  }

  [???]
}

class Dog {
  private boolean isLittle; // is a little dog?

  public Dog(boolean isLittle) {
    this.isLittle = isLittle;
  }

  public boolean isLittle() { return isLittle; }

  public boolean eats(Cat c) {
    return c.isEatenBy(this);
  }
}
public class Exercise {
  public static void main(String[] args) {
    Cat cat = new Cat();
    Dog lilDog = new Dog(true);
    Dog bigDog = new Dog(false);

    assert !lilDog.eats(cat);
    assert bigDog.eats(cat);
  }
}
```

Go back

```
public boolean isEatenBy(Dog dog) {
    if (!dog.isLittle()) {
        return true;
    }
    return false;
}
```

S

SWEN221 2021 T1

Question: Polymorphism Poly7[easy]

Please answer the following question:

```
// The answer must have balanced parentheses
interface Animal {
  public String getName();
  }

class Cat implements Animal {
  public String getName() {
    return "Cat";
  }
}

class Dog implements Animal {
  public String getName() {
    return "Dog";
  }
}

public class Exercise {

  public static String get([???] animal) {
    return animal.getName();
  }

  public static void main(String[] args) {
    Dog dog = new Dog();
    Cat cat = new Cat();
    assert get(dog).equals("Dog");
    assert get(cat).equals("Cat");
  }
}
```

Go back

```
Animal
```

S

```java
// The answer must have balanced parentheses
import java.util.*;

class Cat {
  protected String name;
  public Cat(String name) { this.name = name; }
  public void fight(Cat target, List<String> log) {
    log.add(name + " claws " + target.name);
    target.fightBack(this,log);
  }

  public void fightBack(Cat target, List<String> log) {
    log.add(name + " claws back " + target.name);
  }
}

class RoughCat extends Cat {
  public RoughCat(String name) {
    super(name);
  }

  public void fight(Cat target, List<String> log) {
    log.add(name + " bites " + target.name);
    target.fightBack(this,log);
  }

  public void fightBack(Cat target, List<String> log) {
    log.add(name + " bites back " + target.name);
  }
}

public class Exercise {
  public static void main(String[] args) {
    List<String> log = new ArrayList<String>();
    Cat jim = new Cat("Jim");
    Cat bob = new RoughCat("Bob");
    bob.fight(jim,log);
    assert log.toString().equals([???]);
  }
}
```

```
log.toString()
```

S

```java
// The answer must have balanced parentheses
import java.util.*;

class Cat {
  protected String name;
  public Cat(String name) { this.name = name; }
  public void fight(Cat target, List<String> log) {
    log.add(name + " claws " + target.name);
    target.fightBack(this,log);
  }
  public void fightBack(Cat target, List<String> log) {
    log.add(name + " claws back " + target.name);
  }
}

class RoughCat extends Cat {
  public RoughCat(String name) {
    super(name);
  }

  public void fight(Cat target, List<String> log) {
    log.add(name + " bites " + target.name);
    target.fightBack(this,log);
  }

  public void fightBack(RoughCat target, List<String> log) {
    log.add(name + " bites back " + target.name);
  }
}

public class Exercise {
  public static void main(String[] args) {
    List<String> log = new ArrayList<String>();

    Cat jim = new Cat("Jim");
    Cat bob = new RoughCat("Bob");
    bob.fight(jim,log);

    assert log.toString().equals([???]);
  }
}
```

```
1  log.toString()
```

S

PUZZLERS

SWEN221 2021 T1

Question: Puzzlers IsThisPossible?[hard]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{
  public static void main(String [] arg){
    Integer i=(null);
    String s=(null);
    assert (Object)s== (Object)i;
  }
}
```

Go back

```
null
```

S

SWEN221 2021 T1

Question: Puzzlers PlusFour[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{

  public static int foo=1;

  public static void main(String [] arg){
    int foo=10;
    assert (14==(foo = foo + 4)):"assertion14";
    assert (18==(foo = foo + 4)):"assertion18";
    assert (22==(foo = foo + 4)):"assertion22";
    assert (26==(foo = foo + 4)):"assertion26";
  }
}
```

Go back

```
(foo = foo + 4)
```

S

SWEN221 2021 T1

Question: Puzzlers Range1[easy]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{

  public static void main(String [] arg){
    [???]
    assert (Integer.MAX_VALUE==i);
  }
}
```

Go back

```
int i = 2147483647;
```

S

SWEN221 2021 T1

Question: Puzzlers RangeBlocked[hard]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "Integer" "Float"
public class Exercise{

  public static void main(String [] arg){
    [???]
    assert (Integer.MAX_VALUE==i);
  }
}
```

Go back

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

S

Question: Puzzlers SqlInjection[hard]

Please answer the following question:

```
//The answer is not required to have balanced parentheses inside string l
public class Exercise{

  public static void main(String [] arg){
    int foo=10;
    assert ("Foo".equals("Bar[???]")):"assertionFoo=Bar?";
  }
}
```

Go back

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

S

REFLECTION

SWEN221 2021 T1

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

Please answer the following question:

```
//The answer must have balanced parentheses
class ArrayUtil{

 public static String format(Object[]array,String opening,String closin
   String result=opening;
   if(array.length==0)return result+closing;
   for(int i=0;i<array.length-1;i++){
     result+=array[i]+separator;
   }
   return result+array[array.length-1]+closing;
 }
}
public class Exercise{
 public static void main(String [] arg){
   assert "[1, 2, 3]".equals(ArrayUtil.format(new Integer[]{1,2,3},[???
   assert "[4, 5, 6]".equals(ArrayUtil.format(new Integer[]{4,5,6},[???
   assert "[7, 8, 9]".equals(ArrayUtil.format(new Integer[]{7,8,9},[???
 }
}
```

```
"[" , "]" , ", "
```

S

SWEN221 2021 T1

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

Please answer the following question:

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

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

s

SWEN221 2021 T1

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

Please answer the following question:

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

```
{super.separator = "; ";}
```

S

SWEN221 2021 T1

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

Please answer the following question:

```
//The answer must have balanced parentheses
import java.util.*;
class Point{ int x;int y;
  Point(int x, int y){this.x=x;this.y=y;}
  public String toString(){return "["+x+","+y+"]";}
}
public class Exercise{

  public static void main(String [] arg){
    ArrayList<Point> ps=new ArrayList<Point>();
    ps.add(new Point(2,2));
    ps.add(new Point(1,2));
    ps.add(new Point(3,2));
    ps.add(new Point(2,5));
    Collections.sort(ps,[???]);
    assert(ps.toString().equals("[[1,2], [2,2], [2,5], [3,2]]")):ps.toSt
  }
}
```

Go back

```
new Comparator<Point>() {
    @Override
    public int compare(Point o1, Point o2) {
        if (o1.x < o2.x) {
            return -1;
        } else if (o1.x == o2.x) {
            if (o1.y<o2.y) {
                return -1;
            } else if(o1.y == o2.y) {
                return 0;
            }
        } else {
            return 1;
        }
        return 0;
    }
}
```

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection Instance2[medium]

Please answer the following question:

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

Go back

```
new A(){
    int m() {
        return 2;
    }
}
```

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax1[easy]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{
  static class B{}
  static class C{ B b;
    C(B b){this.b=b;}
    int foo(){return 42;}
  }
}
public class Exercise{

  public static void main(String [] arg){
    assert ([???].foo()==42);
  }
}
```

Go back

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

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax2[easy]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{
  static class B{
    static class C{ B b;
      C(B b){this.b=b;}
      int foo(){return 42;}
    }
  }
}
public class Exercise{

  public static void main(String [] arg){
    assert ([???].foo()==42);
  }
}
```

Go back

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

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax3[medium]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{
  private static class B{}
  static class C{ B b;
    C(B b){this.b=b;}
    int foo(){return 42;}
  }
}
public class Exercise{
  public static void main(String [] arg){
    assert ([???].foo()==42);
  }
}
```

Go back

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

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection NestedSyntax4[hard]

Please answer the following question:

```
//The answer must have balanced parentheses
class A{
  class C{
    int foo(){return 42;}
  }
}
public class Exercise{

  public static void main(String [] arg){
    assert (new A() {
              public int foo(){return 42;}
            }.foo()==42);
  }
}
```

Go back

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

S

SWEN221 2021 T1

Question: Inner/Anon/Reflection Instance1[easy]

Please answer the following question:

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

Go back

INTRODUCTION

SWEN221 2021 T1

Question: Introduction biggerEveryTime[Medium]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{

  public static int foo=1;

  public static void main(String [] arg){
    int foo=10;
    assert (10==[???]):"assertion10";
    assert (11==[???]):"assertion11";
    assert (12==[???]):"assertion12";
    assert (13==[???]):"assertion13";
  }
}
```

Go back

```
1  foo++
```

S

SWEN221 2021 T1

Question: Introduction ChangingTheStart[Medium]

Please answer the following question:

```
//The answer must have balanced parentheses
public class Exercise{

  public static int foo=1;

  public static void main(String [] arg){
    String foo="#";
    assert ("#Bar".equals([???])):"assertion#Bar";
    foo="_";
    assert ("_Bar".equals([???])):"assertion_Bar";
    foo="Bar";
    assert ("BarBar".equals([???])):"assertionBarBar";
  }
}
```

Go back

Photos - two.PNG

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

S

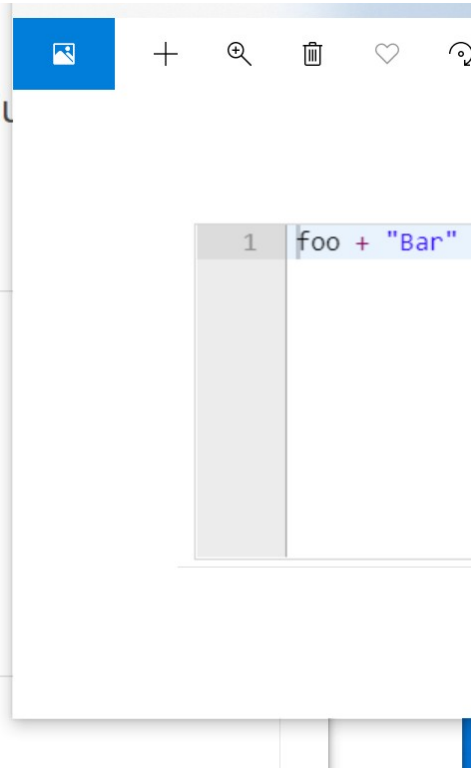
SWEN221 2021 T1

Question: Introduction DoubleEveryTime[Mediu

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
```

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
```

Go back

S

SWEN221 2021 T1

Question: Introduction SearchOnline[medium]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "Float" or "Doubl
public class Exercise{

  public static void main(String [] arg){
    float f=[???];
    assert (Float.MAX_VALUE==f);
  }
}
```

Go back

```
0x1.fffffeP+127f
```

S

SWEN221 2021 T1

Question: Introduction Simple[easy]

Please answer the following question:

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

Go back

```
10
```

S

SWEN221 2021 T1

Question: Introduction Simple?[hard]

Please answer the following question:

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

Go back

```
5
```

s

SWEN221 2021 T1

Question: Introduction +Infinity[medium]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "Float" or "Doubl
public class Exercise{

  public static void main(String [] arg){
    assert (Float.POSITIVE_INFINITY==[???]);
  }
}
```

Go back

```
1.0f / 0.0f
```

s

SWEN221 2021 T1

Question: Introduction -Infinity[medium]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "Float" or "Doubl
public class Exercise{

  public static void main(String [] arg){
    assert (Float.NEGATIVE_INFINITY==[???]);
  }
}
```

Go back

```
-1.0f / 0.0f
```

S

SWEN221 2021 T1

Question: Introduction FloatsAreSurprising[hard]

Please answer the following question:

```
//The answer must have balanced parentheses and not use "Float" or "Doubl
public class Exercise{

  public static void main(String [] arg){
    float f=[???];
    assert (f!=f);
  }
}
```

Go back

```
0.0f / 0.0f
```

S

SWEN221 2021 T1

Question: Introduction MyMax[easy]

Please answer the following question:

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

```
float max = elems.get(0);
for (int i = 0; i < elems.size(); ++i) {
    if (elems.get(i) > max) {
        max = elems.get(i);
    }
}
return max;
```