Aufgabe Fehleranalyse (Kotlin-Variante): Gegeben seien folgende Schnittstellen und Klassen.

```
interface Ausgabe {
  fun ausgabe()
}
open class Haus : Ausgabe {
  override fun ausgabe() {  print("Haus") }
}
class Villa : Haus() {
  override fun ausgabe() { print("Villa")  }
}
class BaumHaus : Haus() {
   fun baumtyp() { print("Palme") }
}
```

Schreiben Sie hinter alle Zeilen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler (CF) oder einen Laufzeitfehler (LF) verursacht:

```
val h1 : Haus = Haus()
val h2 : Haus = Villa()
val h3 : Haus = BaumHaus()
val a1 : Ausgabe = Ausgabe()
val a2 : Ausgabe = Haus()
val a3 : Ausgabe = a2
val a4 : Ausgabe = h1
val v1 : Villa = h1
val v2 : Villa = h2
val v3 : Villa = h2 as Villa
val v4 : Villa = h3 as Villa
val h4 : Haus = v3
val v5 : Villa = h2
```

Welcher Text wird ausgegeben („Haus“, „Villa“, „Palme“)? Wo gibt es einen Compilerfehler?

```
h1.ausgabe()
h2.ausgabe()
h3.ausgabe()
h3.baumtyp()
a2.ausgabe()
a4.ausgabe()
v3.ausgabe()
h4.ausgabe()
h4.jetztEinziehen()
```

Aufgabe Fehleranalyse (JAVA-Variante): Gegeben seien folgende Schnittstellen und Klassen.

```
public interface Ausgabe { public void ausgabe () ; }

public class Haus implements Ausgabe {

public void ausgabe() { System.out.print("Haus"); } }

public class Villa extends Haus {

public void ausgabe() { System.out.print("Villa"); } }

public class BaumHaus extends Haus {

public void baumtyp() { System.out.print("Palme"); } }
```

Schreiben Sie hinter alle Zeilen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler (CF) oder einen Laufzeitfehler (LF) verursacht:

```
        Haus h1 = new Haus ();
        Haus h2 = new Villa();
        Haus h3 = new Baumhaus ();
        Ausgabe a1 = new Ausgabe();
        Ausgabe a2 = new Haus();
        Ausgabe a3 = a2;
        Ausgabe a4 = h1;
        Villa v1 = h1;
        Villa v2 = h2;
        Villa v3 = (Villa) h2;
        Villa v4 = (Villa) h3;
        Haus h4 = v3;

Villa v5 = h4;
```

Welcher Text wird ausgegeben („Haus“, „Villa“, „Palme“)? Wo gibt es einen Compilerfehler?

```
h1.ausgabe(); h2.ausgabe(); h3.ausgabe();
h3.baumtyp(); a2.ausgabe(); a4.ausgabe();
v3.ausgabe(); h4.ausgabe(); h4.jetztEinziehen(); ____
```

Aufgabe Fehleranalyse (Kotlin-Variante): Gegeben seien folgende Schnittstellen und Klassen.

```
interface Speaker {  fun say() }
interface Dancer {   fun party() }
open class Person : Speaker, Dancer {
    override fun say()   { print("Hello")          }
    override fun party() { print("Endless Summer") }
}
class Student : Person() {
    override fun say() {

super.say()

        print("Think")
    }

}
```

Schreiben Sie hinter alle Zeilen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler (CF) oder einen Laufzeitfehler (LF) verursacht:

```
val p1 : Person = Person()
val s : Student = Student()
val speaker1 : Speaker = p1
val speaker2 : Speaker = s
val speaker3 = Speaker()
val d1 : Dancer = Person()
val d2 : Dancer = s as Dancer
val d3 : Dancer = p1
val d4 : Dancer = p1 as Speaker
val p2 : Person = p1
val p3 : Person= p1
val p4 : Person= s as Person
val s2 : Student = p1 as Student
```

Welcher Text wird ausgegeben?

```
p1.say()
(s as Person).say()
s.say()
p2.say()
p4.say()
speaker1.say()
p1.party()
Student().party()
```

Aufgabe Fehleranalyse (JAVA-Variante): Gegeben seien folgende Schnittstellen und Klassen.

```
public interface Speaker { public void say () ; }

public interface Fun { public void party () ; }

public class Person implements Speaker, Fun {
public void say() { System.out.print("Hello"); }

public void party() { System.out.print("Endless Summer"); } }

public class Student extends Person {

public void say () {
super.say();

System.out.print("Think"); }

}
```

Schreiben Sie hinter alle Zeilen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler (CF) oder einen Laufzeitfehler (LF) verursacht:

```
Person p1 = new Person ();
Student s = new Student();
Speaker speaker1 = p1;
Speaker speaker2 = s;
Speaker speaker3 = new Speaker ();
Fun f1 = new Person();
Fun f2 = (Fun) s;
Fun f3 = speaker1;
Fun f4 = (Speaker) p1;
Person p2 = p1;
Person p3 = speaker1;
Person p4 = (Person) speaker2;
Student s2 = (Student) p1;
```

Die folgenden Zeilen sind fehlerfrei. Welcher Text wird auf der Konsole ausgegeben?

```
p1.say();
((Person) s).say();
s.say();
p2.say();
p4.say();
speaker1.say();
p1.party();
new Student().party();
```

Aufgabe Fehleranalyse: Gegeben seien folgende Schnittstellen und Klassen.

```
abstract public class Fahrzeug {
public int geschwindigkeit;
public int beschleunigen () { return geschwindigkeit++; }
public int bremsen () { return geschwindigkeit--; }

  abstract public void fahren (); public class Auto extends
}
Fahrzeug {

  public void fahren() { beschleunigen(); beschleunigen(); }
}

public class Motorrad extends Fahrzeug {
private int gefahreneStrecke = 0;
public void fahren() { gefahreneStrecke += geschwindigkeit ; }

  public void ausgabe() { System.out.println
}
(gefahreneStrecke); }
```

Schreiben Sie hinter alle Zeilen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler (CF) oder einen Laufzeitfehler (LF) verursacht. Schreiben Sie ggf. auf, welche Ausgabe auf der Konsole mit `System.out.println(...)` **erzeugt wird.**

```
Fahrzeug f1 = new Fahrzeug();
Fahrzeug f2 = new Auto ();
Fahrzeug f3 = new Motorrad ();
Motorrad m1 = new Motorrad ();
Auto a = (Auto) f2;
Motorrad m2 = m1;
Motorrad m3 = f3;
Motorrad m4 = (Motorrad) f2;
Motorrad m5 = (Motorrad) f3;
f2.beschleunigen();
f2.fahren();
f2.ausgabe()
m5.ausgabe()
a.beschleunigen();
a.fahren();
f3.gibGefahreneStreckeAus();
m5.gibGefahreneStreckeAus();
System.out.println( m5.gefahrenStrecke  );
System.out.println( a.geschwindigkeit );
System.out.println( m5.geschwindigkeit );
System.out.println( m5.beschleunigen());
System.out.println( m5.bremsen());
```

Aufgabe Fehleranalyse: Gegeben seien folgende Schnittstellen und Klassen.

```
public interface LaesstSichWiegen {
     public int getGewicht ();
}
abstract public class Obst {
     public int gewicht;
     public int getGewicht () {  return gewicht; }

}

public class Birne extends Obst {
     public void setGewicht (int g) {
         this.gewicht = g;
     }
}
public class Apfel extends Obst implements LaesstSichWiegen {
}
```

Schreiben Sie hinter alle Zeilen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler (CF) oder einen Laufzeitfehler (LF) verursacht.

```
Obst o = new Obst ();
Apfel a = new Apfel ();
Birne b = new Birne ();
Obst o2 = new Apfel();
Apfel a2 = o2;
Apfel a3 = (Apfel) o2;
Birne b2 = o2;
Birne b3 = (Birne) o2;
LaesstSichWiegen w1 = o2;
LaesstSichWiegen w2 = a3;
LaesstSichWiegen w3 = b3;
System.out.println( a.getGewicht() );
System.out.println( b.getGewicht() );
System.out.println( o2.getGewicht() );
System.out.println( w2.getGewicht() );
a.setGewicht(30);
b.setGewicht(55);
a.gewicht = 14;
o2.gewicht = 22;
w2.gewicht = 42;
```

Aufgabe Feheranalyse: Gegeben seien folgende Schnittstelle und Klassen:

```
public interface MagicNumber {

     int getNextNumber () ;
}

public class CountUp implements MagicNumber { private int counter = 0;

@Override public int getNextNumber() { return counter++; } }

public class CountDown implements MagicNumber { private int counter = 0;

@Override public int getNextNumber() { return counter--; } }

public class Constant implements MagicNumber { private int value;

public Constant(int value) { this.value = value; }
  @Override public int getNextNumber() {return value; }
}
```

Schreiben Sie hinter alle Zuweisungen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler oder einen Laufzeitfehler verursacht.
Bei `println(...)` **Anweisungen schreiben Sie bitte die jeweilige Ausgabe dahinter.**

```
MagicNumber a = new Constant (33);
Constant b = new Constant (44);
CountUp c = new CountUp();
MagicNumber d = new CountDown();
System.out.println (c.getNextNumber());
System.out.println (c.getNextNumber());
d = a;
a = d;
d = c;
System.out.println (a.getNextNumber());
System.out.println (b.getNextNumber());
System.out.println (c.getNextNumber());
System.out.println (d.getNextNumber());
a = (MagicNumber) b;
a = (Constant) b;
b = (Constant) a;
b = (Constant) c;
b = (Constant) d;
c = a;
c = b;
c = d;
a = a;
a = b;
a = c;
a = d;
```

Aufgabe Fehleranalyse: Gegeben seien folgende Klassen:

```
public abstract class Tier {
private Tier gefuerchtet; // jenes Tier wird von diesem Tier gefürchtet
public String name; // Name dieses Tiers

public Tier(Tier gefuerchtet , String name) {

this.gefuerchtet = gefuerchtet; this.name = name; }

abstract public void sound ();

public void hatAngstVor() { System.out.println (gefuerchtet.name); } }

public class Katze extends Tier {
public Katze(Tier gefuerchtet, String name) { super(gefuerchtet, name); }

@Override public void sound() { System.out.println ("Miau!"); } }

public class Maus extends Tier{
public Maus(Tier gefuerchtet, String name) { super(gefuerchtet, name); }

@Override public void sound() { System.out.println ("Pieps!"); } }

public class Elefant extends Tier{
public Elefant(Tier gefuerchtet, String name) { super(gefuerchtet, name);}

@Override public void sound() { System.out.println("Töröööö!"); } }
```

Schreiben Sie hinter alle Zeilen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler (CF) oder einen Laufzeitfehler (LF) verursacht. Beim Aufruf von `hatAngstVor(...)` **und** `sound()` **schreiben Sie bitte die jeweilige Ausgabe dahinter.**

```
Katze k = new Katze (null , "Garfield");
Elefant e=new Elefant (new Maus (k,"Micky") , "Benjamin"); Elefant e2 = new
Tier ( k , "Dumbo");
Tier m = new Maus ( k , "Tom");
Tier t = new Tier (m , "Alf");
e.hatAngstVor();
m.hatAngstVor();
k.hatAngstVor();
Tier tKatze = k;
Tier tElefant = (Tier) e;
tElefant.hatAngstVor();
tKatze.sound();
tElefant.sound();
k.name = "Jerry";
m.name = k.name;
k = (Katze) tKatze;
k = (Katze) tElefant;
m.hatAngstVor();
e.hatAngstVor();
e = new Maus ( k , "Speedy");
```

Aufgabe Fehleranalyse: Gegeben seien folgende Schnittstelle und Klassen:

```
public interface MagicWord {
    String getNextWord();
}
public class SingleWord implements MagicWord{

private String word;
public SingleWord (String w) { word = w; }

@Override public String getNextWord() { return word; } }

public class MultipleWords implements MagicWord {
private String [] words;
private int index;
public MultipleWords(String [] words) { this.words = words; }

     @Override
     public String getNextWord() {
     return words [index++  %  words.length];
     }

}
```

Schreiben Sie hinter alle Zuweisungen, ob die Anweisung fehlerfrei ist (OK), einen Compilerfehler oder einen Laufzeitfehler verursacht.
Bei `println(...)` **Anweisungen schreiben Sie bitte die jeweilige Ausgabe dahinter.**

```
String [] f = {"rot", "grün", "blau"};
String [] c = {"Hamburg", "Berlin", "Köln" , "München"}; MagicWord farben =
new MultipleWords ( f );
MultipleWords staedte = new MultipleWords (c);
MagicWord hamburg = new SingleWord (staedte.getNextWord()); SingleWord
hallo = new SingleWord ("hallo");
System.out.println(hamburg.getNextWord());
System.out.println(staedte.getNextWord());
System.out.println(staedte.getNextWord());
System.out.println(hamburg.getNextWord());
farben = staedte;
staedte = farben;
hamburg = hallo;
System.out.println(hamburg.getNextWord());
System.out.println(farben.getNextWord());
System.out.println(farben.getNextWord());
System.out.println(hamburg.getNextWord());
farben = hamburg;
staedte = (MultipleWords) farben;
staedte = (SingleWord) farben;
hallo = (MultipleWords) farben;
hallo = (SingleWord) farben;
```