Lambda Fragen

4. Given:

```
import java.util.*;
import java.util.function.Predicate;
public class Birds {
  Birds(int w, boolean talk) { ounceWeight = w; canTalk = talk; }
  int ounceWeight;
  boolean canTalk;
  int getW() { return ounceWeight; }
  boolean getTalk() { return canTalk; }
  public String toString() { return "" + getW() + " " + getTalk(); }
  public static void main(String[] args) {
    ArrayList<Birds> birds = new ArrayList<>();
    birds.add(new Birds(1, true));
    birds.add(new Birds(1, false));
    birds.add(new Birds(48, false));
    birds.add(new Birds(32, true));
    System.out.println("parrots: "
      + sorter(birds, b -> b.getTalk() == true));
    System.out.println("  parrotlets: "
      + sorter(birds, b -> b.getTalk() == true &&
                           b.getW() <= 2));
  }
  static ArrayList<Birds> sorter(ArrayList<Birds> blist,
                                 Predicate<Birds> expr) {
    ArrayList<Birds> result = new ArrayList<>();
    for(Birds b: blist)
      if(expr.test(b))
        result.add(b);
    return result;
  }
}
```

What could be the result?

A. parrots: [1 true]
   parrotlets: [1 true]

B. parrots: [Birds@1fb3ebeb]
   parrotlets: [Birds@1fb3ebeb]

C. parrots: [Birds@1fb3ebeb, Birds@548c4f57]
   parrotlets: [Birds@1fb3ebeb]

D. parrots: [1 true, 32 true]
   parrotlets: [1 true]

E. Compilation fails

F. An exception is thrown at runtime

13. Given:

```
import java.util.function.Lambda;
public class Java8 {
  static boolean b;
  public static void main(String[] args) {
    Java8 s = new Java8();
    s.go(x -> mult(5, 1) < 7);
    s.go(x -> b = new Boolean(true));
    s.go(y -> { return mult(3, 2) < 4; });
  }
  void go(Predicate<Java8> e) {
    Java8 s2 = new Java8();
    if(e.test(s2))
      System.out.print("true ");
    else
      System.out.print("false ");
  }
  static int mult(int x, int y) {
    return x * y;
  }
}
```

wrong import ;)

What is the result?

- A. true true false
- B. true false true
- C. true true true
- D. Compilation fails
- E. An exception is thrown at runtime

14. Given:

```
import java.util.function.Predicate;
public class EvenMoreSheep {
  static boolean b = false;
  public static void main(String[] args) {
    EvenMoreSheep s = new EvenMoreSheep();
    EvenMoreSheep s2 = s;
    EvenMoreSheep s3;
    s.go(x -> b == false);          // line A
    s.go(x -> s == s2);             // line b
    s.go(x -> s3 = s);              // line c
  }
  void go(Predicate<EvenMoreSheep> e) {
    EvenMoreSheep s2 = new EvenMoreSheep();
    if(e.test(s2))
      System.out.print("true ");
    else
      System.out.print("false ");
  }
  static int adder(int x, int y) {
    return x + y;
  }
}
```

What is the result? (Choose all that apply.)

- [ ] A. true true true
- [ ] B. true true false
- [ ] C. true false false
- [ ] D. Line a will not compile
- [ ] E. Line b will not compile
- [ ] F. Line c will not compile

zuweisung !! kein Vergleich --> Kein boolsches Ergebnis

17. What's true about the `Predicate` interface? (Choose all that apply.)

- [ ] A. It has only one method
- [ ] B. To use `Predicate`, you typically invoke its `ifTrue()` method
- [x] C. It has only one non-concrete method
- [x] D. The non-concrete method(s) you use is/are invoked using a `Predicate`
- [ ] E. The non-concrete method(s) you use return(s) a `String`
- [ ] F. The `Predicate` interface is in the `java.util.lambda` package

A,B,E,and F are incorrect. A is incorrect because Predicate has several default and static methods. B is incorrect because you invoke its test() method. E is incorrect because test() returns a boolean. F is incorrect because the package is java.until.function

20. Given:

```
import java.util.function.Predicate;
public class MoreSheep {
  static boolean b = false;
  public static void main(String[] args) {
    MoreSheep s = new MoreSheep();
    s.go(x -> b == false);          // line A
    s.go(x -> adder(4, 2) >= 6);    // line B
    s.go(x -> b = true);            // line C
    s.go(x -> adder(3, 2) < 4);     // line D
  }
  void go(Predicate<MoreSheep> e) {
    MoreSheep s2 = new MoreSheep();
    if(e.test(s2))
      System.out.print("true ");
    else
      System.out.print("false ");
  }
  static int adder(int x, int y) {
    return x + y;
  }
}
```

only comparing line with a boolean

What is the result?

- A. true true true false
- B. true true false false
- C. false true true false
- D. false true false false
- E. There is at least one compiler error in Lines A–D
- F. An exception is thrown at runtime

Person.java:

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

    public Person(String n, int a) {
        name = n;
        age = a;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }
}
```

Test.java:

```
public static void checkAge(List<Person> list, Predicate<Person> predicate) {
    for (Person p : list) {
        if (predicate.test(p)) {
            System.out.println(p.name + " ");
        }
    }
}

public static void main(String[] args) {
    List<Person> iList = Arrays.asList(new Person("Hank", 45),
                                       new Person("Charlie", 40),
                                       new Person("Smith", 38));
    //line n1
}
```

Which code fragment, when inserted at line n1, enables the code to print Hank?

A. checkAge (iList, ( ) -> p. get Age ( ) > 40); zweite Parameter fehlt
B. checkAge(iList, Person p -> p.getAge( ) > 40); zweite parameter ist Predicate NICHT Person
C. checkAge (iList, p -> p.getAge ( ) > 40);
D. checkAge(iList, (Person p) -> { p.getAge() > 40; }); selbe wie in B

QUESTION 72
Given the code fragment:

```
 3. public static void main(String[] args) {
 4.       int x = 5;
 5.       while (isAvailable(x)) {
 6.           System.out.print(x);
 7.
 8.       }
 9. }
10.
11. public static boolean isAvailable(int x) {
12.     return x-- > 0 ? true : false;
13. }
```

Dekrement Operator fehlt ; Hat nüscht mit Lambda zu tun ;)

Which modification enables the code to print 54321?

A. Replace line 6 with System, out. print (--x) ;
B. At line 7, insert x --;
C. Replace line 6 with –x; and, at line 7, insert system, out. print (x);
D. Replace line 12 With return (x > 0) ? false: true;

QUESTION 150
Given:

```
1. import java.util.ArrayList;
2. import java.util.List;
3.
4. public class Whizlabs{
5.
6.          public static void main(String[] args){
7.                    List<Integer> list = new ArrayList<>();
8.                    list.add(21); list.add(13);
9.                    list.add(30); list.add(11);
10.                   list.add(2);
11.                   //insert here
12.                   System.out.println(list);
13.         }
14. }
```

Which inserted at line 11, will provide the following output?

```
[21, 13, 11]
```

A. list.removeIf(e > e%2 != 0);
B. list.removeIf(e -> e%2 != 0);
C. list.removeIf(e->e%2=0);
D. list.remove(e -> e%2 = 0);
E. None of the above.

QUESTION 194
Consider following interface.

```
interface Runnable{
        public void run();
}
```

Which of the following will create instance of Runnable type?

A. Runnable run = 0 -> {System.out.println("Run");}
B. Runnable run = 0 -> System.outprintlnfRun");
C. Runnable run = 0 > System.outprintlnfRun");
D. Runnable run = > System.ouLprintlnfRun"};
E. None of the above.

QUESTION 206
Given:

```
1. import java.util.ArrayList;
2. import java.util.List;
3.
4. public class Whizlabs{
5.
6.          public static void main(String[] args){
7.                  List<int> list = new ArrayList<>();
8.                  list.add(21); list.add(13);
9.                  list.add(30); list.add(11);
10.                 list.removeIf(e -> e%2 != 0);
11.                 System.out.println(list);
12.         }
13. }
```

What is the output?

A. [21, 13, 11]
B. [30]
C. []
D. Compilation fails due to error at line 7
E. Compilation tails due to error at line 10

primitive type

13. Given:

```
import java.util.function.Predicate;
public class Sheep {
  public static void main(String[] args) {
    Sheep s = new Sheep();
    s.go(() -> adder(5, 1) < 7);     // line A
    s.go(x -> adder(6, 2) < 9);      // line B
    s.go(x, y -> adder(3, 2) < 4);   // line C
  }
  void go(Predicate<Sheep> e) {
    Sheep s2 = new Sheep();
    if(e.test(s2))
      System.out.print("true ");
    else
      System.out.print("false ");
  }
  static int adder(int x, int y) {
    return x + y;
  }
}
```

What is the result?

A. `true true false`
B. Compilation fails due only to an error at line A
C. Compilation fails due only to an error at line B
D. Compilation fails due only to an error at line C
E. Compilation fails due only to errors at lines A and B
F. Compilation fails due only to errors at lines A and C
G. Compilation fails due only to errors at lines A, B, and C
H. Compilation fails for reasons not listed

16. Given that `adder()` returns an `int`, which are valid `Predicate` lambdas? (Choose all that apply.)

A.
```
x, y -> 7 < 5
```

B.
```
x -> { return adder(2, 1) > 5; }
```

C.
```
x -> return adder(2, 1) > 5;
```

D.
```
x -> { int y = 5;
       int z = 7;
       adder(y, z) > 8; }
```

E.
```
x -> { int y = 5;
       int z = 7;
       return adder(y, z) > 8; }
```

F.
```
(MyClass x) -> 7 > 13
```

G.
```
(MyClass x) -> 5 + 4
```

19. Given that `e` refers to an object that implements `Predicate`, which could be valid code snippets or statements? (Choose all that apply.)

A.
```
if(e.test(m))
```

B.
```
switch (e.test(m))
```
switch kann KEIN boolean

C.
```
while(e.test(m))
```

D.
```
e.test(m) ? "yes" : "no";
```

E.
```
do {} while(e.test(m));
```

F.
```
System.out.print(e.test(m));
```

G.
```
boolean b = e.test(m));
```