

# Lecture 15

Review

`\b[A-Z0-9._%+-]+\@[A-Z0-9.-]+\.[A-Z]{2,}\b`

What does this match?

Matching  
special  
characters  
using  
backslash to  
escape

```
1 import java.util.*;
2 import java.util.regex.*;
3
4 public class PatternQuoteExample {
5     public static void main (String[] args) {
6         String input = "Math operators: +-* /. ";
7         boolean result;
8
9         String quoted = Pattern.quote("+-* /.");
10        System.out.println(quoted);
11
12        // regex using standard escaping
13        result = input.matches(".*\\s+\\+\\-\\*\\/\\.\\s+.*");
14        System.out.println(result);
15
16        // regex Using Pattern.quote around our search string
17        result = input.matches(".*\\s+" + quoted + "\\s+.*");
18        System.out.println(result);
19
20        // regex Using \\Q and \\E around our search string
21        result = input.matches(".*\\s+\\Q+-*/.\\E\\s+.*");
22        System.out.println(result);
23    }
24 }
```

```
\Q+-*/. \E
true
true
true
```

```
1 import java.util.*;
2 import java.util.regex.*;
3 public class PatternSplitExample {
4     public static void main (String[] args) {
5         final String input = "value1||value2||value3";
6         final Pattern p = Pattern.compile( Pattern.quote( "||" ) );
7
8         // call split and print each element from generated array
9         // using stream API
10        Arrays.stream( p.split(input) ) // p.splitAsStream(input)
11            .forEach( System.out::println );
12    }
13 }
```

```
H:\work\CS209A_19S\notes08>javac PatternSplitExample.java
```

```
H:\work\CS209A_19S\notes08>java PatternSplitExample
```

```
value1
```

```
value2
```

```
value3
```

Splitting using regular expressions instead of string operations

```
1 import java.util.List;
2 import java.util.stream.*;
3 import java.util.regex.*;
4
5 public class AsPredicateExample {
6     public static void main (String[] args) {
7         final String[] monthsArr =
8             { "10", "0", "05", "09", "12", "15", "00", "-1", "100" };
9         final Pattern validMonthPattern =
10             Pattern.compile( "^(?:0?[1-9]|1[0-2])$" );
11         List<String> filteredMonths =
12             Stream.of( monthsArr )
13                 .filter( validMonthPattern.asPredicate() )
14                 .collect( Collectors.toList() );
15         System.out.println( filteredMonths );
16     }
17 }
```

```
H:\work\CS209A_19S\notes08>javac AsPredicateExample.java
```

```
H:\work\CS209A_19S\notes08>java AsPredicateExample
[10, 05, 09, 12]
```

```

1 import java.util.Arrays;
2 import java.util.List;
3 import java.util.function.Predicate;
4 import java.util.regex.Pattern;
5 import java.util.stream.Collectors;
6
7 public class RegexPredicateExample {
8     public static void main(String[] args) {
9         // Compile regex as predicate
10        Predicate<String> emailFilter =
11            Pattern.compile("^(.+)@example.com$")
12                .asPredicate();
13
14        // Input list
15        List<String> emails = Arrays.asList(
16            "alex@example.com", "bob@yahoo.com",
17            "cat@google.com", "david@example.com"
18        );
19
20        // Apply predicate filter
21        List<String> desiredEmails =
22            emails.stream()
23                .filter(emailFilter)
24                .collect(Collectors.<String>toList());
25
26        // Now perform desired operation
27        desiredEmails.forEach(System.out::println);
28    }
29 }

```

Making a predicate and using stream processing supported by regex applied to filter emails by domain

```
alex@example.com
david@example.com
```

```

1  import java.util.regex.*;
2
3  public class MatcherMatchesExample {
4      public static void main (String[] args) {
5          final Pattern pattern1 = Pattern.compile( "mastering" );
6          final Pattern pattern2 = Pattern.compile( "mastering.*" );
7          final Pattern pattern3 = Pattern.compile( "regular.*" );
8
9          String input = "mastering regular expressions";
10         Matcher matcher = pattern1.matcher(input);
11         System.out.printf( "[%s] => [%s]: %s%n",
12             input, matcher.pattern(), matcher.matches());
13
14         // update the matcher pattern with a new pattern
15         matcher.usePattern(pattern2);
16         System.out.printf( "[%s] => [%s]: %s%n",
17             input, matcher.pattern(), matcher.matches());
18
19         // update the matcher pattern with a new pattern
20         matcher.usePattern(pattern3);
21         System.out.printf( "[%s] => [%s]: %s%n",
22             input, matcher.pattern(), matcher.matches());
23     }
24 }

```

```

H:\work\CS209A_19S\notes08>java MatcherMatchesExample
[mastering regular expressions] => [mastering]: false
[mastering regular expressions] => [mastering.*]: true
[mastering regular expressions] => [regular.*]: false

```

Using matching (we set a different pattern in java regex matcher class)

Each pattern may or may not match the string given as input ("mastering regular expressions")

```

1 import java.util.regex.*;
2 public class MatcherFindExample {
3     public static void main (String[] args) {
4         final String input =
5             "some text <value1> anything <value2><value3> here";
6
7         /* Part 1 */
8         final Pattern pattern = Pattern.compile( "<([^<>]*)>" );
9         Matcher matcher = pattern.matcher(input);
10        while (matcher.find()) {
11            System.out.printf( "[%d] => [%s]%n",
12                matcher.groupCount(), matcher.group(1) );
13        }
14
15        /* Part 2 */
16        // now use similar pattern but use a named group and reset the
17        // matcher
18        matcher.usePattern( Pattern.compile( "<(?!<name>[^<>]*)>" ) );
19        matcher.reset();
20        while (matcher.find()) {
21            System.out.printf( "[%d] => [%s]%n",
22                matcher.groupCount(), matcher.group("name"));
23        }
24    }
25 }

```

```

[1] => [value1]
[1] => [value2]
[1] => [value3]
[1] => [value1]
[1] => [value2]
[1] => [value3]

```

A template to match some elements of a string delineated by angle brackets.

The substrings that are extracted/matched are able to be obtained from the matcher group (they are numbered from 1 to the number of matches).

Part 2 shows the use of a named group instead of a numbered group (easier to read).



```
H:\work\CS209A_19S\notes08>java MatcherAppendExample  
<v1=n1 v2=n2 v3=n3> n1=v1 n2=v2 abc=123 <pq=v abc=id> v=pq
```

```
1 import java.util.regex.*;  
2  
3 public class MatcherAppendExample {  
4     public static void main (String[] args) {  
5         final String input =  
6             "<n1=v1 n2=v2 n3=v3> n1=v1 n2=v2 abc=123 <v=pq id=abc> v=pq";  
7  
8         // pattern1 to find all matches between < and >  
9         final Pattern pattern = Pattern.compile( "<[^>]+>" );  
10  
11        // pattern1 to find each name=value pair  
12        final Pattern pairPattern = Pattern.compile( "(\\w+)=(\\w+)" );  
13        Matcher enclosedPairs = pattern.matcher(input);  
14  
15        StringBuffer sbuf = new StringBuffer();  
16        // call find in a loop and call appendReplacement for each match  
17        while (enclosedPairs.find()) {  
18            Matcher pairMatcher = pairPattern.matcher( enclosedPairs.group());  
19            // replace name=value with value=name in each match  
20            enclosedPairs.appendReplacement(  
21                sbuf, pairMatcher.replaceAll( "$2=$1" )  
22            );  
23        }  
24        // appendTail to append remaining character to buffer  
25        enclosedPairs.appendTail( sbuf );  
26        System.out.println( sbuf );  
27    }  
28 }
```

Finding multiple matches with a loop (we don't have to know in advance how many matches there will be).

```
java.net.URL url = new URL( sURL );
```

```
106 /**
107  * Initializes an input stream from a URL.
108  *
109  * @param url the URL
110  * @throws IllegalArgumentException if cannot open {@code url}
111  * @throws IllegalArgumentException if {@code url} is {@code null}
112  */
113 public In(URL url) {
114     if (url == null) throw new IllegalArgumentException("url argument is null");
115     try {
116         URLConnection site = url.openConnection();
117         InputStream is      = site.getInputStream();
118         scanner              = new Scanner(new BufferedInputStream(is), CHARSET_NAME);
119         scanner.useLocale(LOCALE);
120     }
121     catch (IOException ioe) {
122         throw new IllegalArgumentException("Could not open " + url, ioe);
123     }
124 }
```

Regex are widely used in network programming.

# Exam Review

# Exam format

- Multiple choice (approximately 50% of the marks)
- Further questions which require you to write an answer where you may be asked to write code

# Likely to have questions on...

- Writing classes (please see the examples this week in the videos also)
- Regular expressions
- Object oriented programming
- Exception handling
- Reflection
- Network programming
- Databases (JDBC)
- Other topics presented in the slides

Sample Exam Question:

# **Huge** Integers

# Huge Integers

In Java a `long` can have a value that is at most  $2^{63}-1$ :

9,223,372,036,854,775,807

Yes its big, but might not be big enough for everyone...

# Multiples of 103 (polynomial approach of the problem)...

123,456,789,012,345,678,901 can be written as:

$$\begin{aligned} &123 \times 1000^6 \\ &+ 456 \times 1000^5 \\ &+ 789 \times 1000^4 \\ &+ 12 \times 1000^3 \\ &+ 345 \times 1000^2 \\ &+ 678 \times 1000^1 \\ &+ 901 \times 1000^0 \end{aligned}$$

## Question – part 1

Of the interfaces available in Java collections (List, Queue/Deque, Set), to which we can add the Map interface, which one seems to you the most appropriate to store HugelInteger values?



# Answer: Which interface??

One value → ~~Map~~

Same number can appear several times → ~~Set~~

A List or Queue is better

# using pseudocode...

**Question – part 2** Write (pseudo) code for the two following constructors.

`HugeInteger(long intValue)`

`HugeInteger(String strValue)`

The string can contain commas to separate thousands or not. Define exceptions and create specific exceptions if needed, or you may throw any of the following existing exceptions if appropriate:

`ArithmeticException` `ArrayIndexOutOfBoundsException`

`IllegalArgumentException` `IndexOutOfBoundsException`

`NegativeArraySizeException` `NullPointerException` `NumberFormatException`

`StringIndexOutOfBoundsException` `UnsupportedOperationException`

# HugeInteger(long intValue)

```
public HugeInteger(long intValue)
    List list <- []
    do:
        append (intValue % 1000) to the list
        intValue = intValue / 1000

    while (intValue > 1)
```

# HugeInteger(String strValue)

```
public HugeInteger(String strValue) throws NumberFormatException {
```

```
    String str = strValue.replace(",", "");
```

```
    while length of strValue > 3 {
```

```
        Take the 3 last characters
```

```
        Convert to integer using Integer.parseInt
```

```
        add value to list
```

```
        set strValue to substring that excludes the last 3 characters
```

```
    }
```

```
    if length(strValue) > 0 {
```

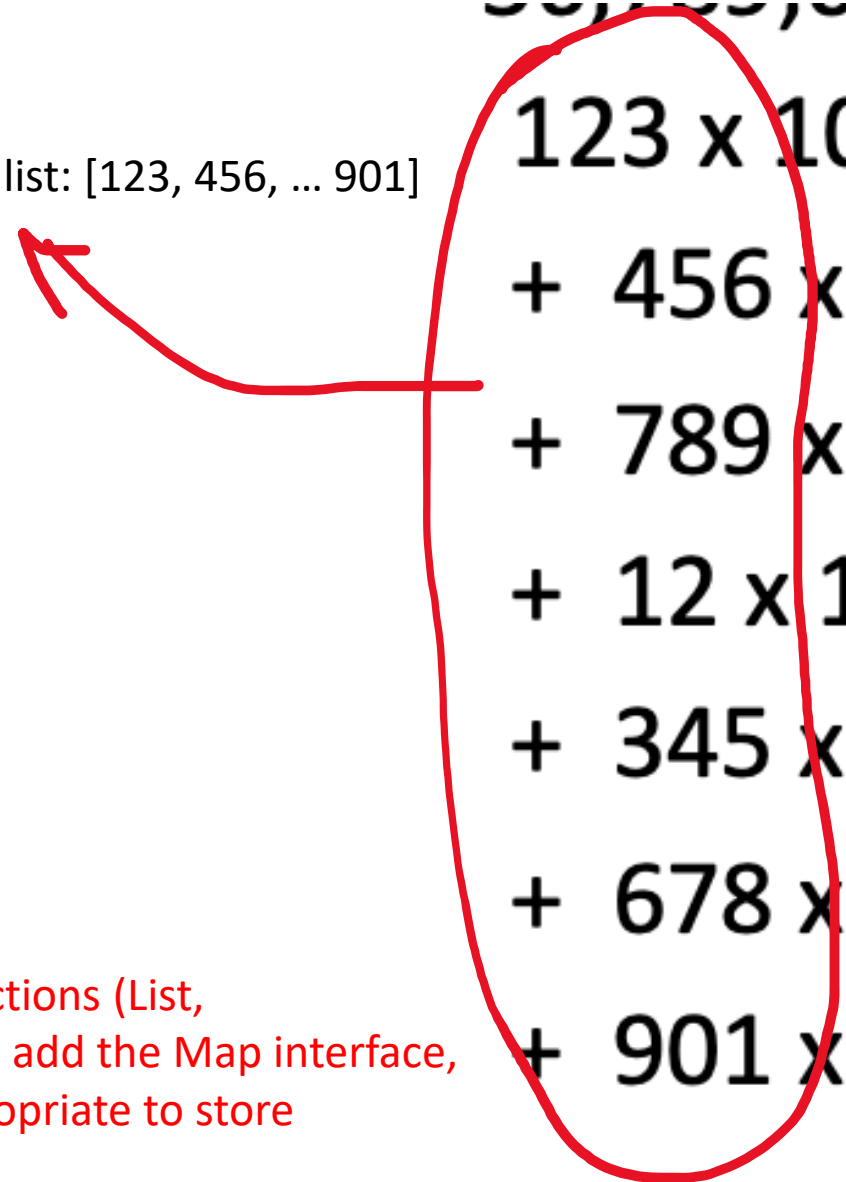
```
        Convert to integer
```

```
        add value to list
```

```
    }
```

Could also use regular expressions  
to split/ tokenize the string on  
“”...

to the list: [123, 456, ... 901]


$$\begin{aligned} &123 \times 1000^6 \\ &+ 456 \times 1000^5 \\ &+ 789 \times 1000^4 \\ &+ 12 \times 1000^3 \\ &+ 345 \times 1000^2 \\ &+ 678 \times 1000^1 \\ &+ 901 \times 1000^0 \end{aligned}$$

Of the interfaces available in Java collections (List, Queue/Deque, Set), to which we can add the Map interface, which one seems to you the most appropriate to store HugeInteger values?

which one seems to you the most appropriate to store HugelInteger values?

- One value -> ~~Map~~
- Same number several times? -> ~~Set~~
- A list or queue perhaps?

## Question – part 3

Define an add method to add two HugelInteger objects. Please note that with regular addition the variables that are added are left unmodified (if you have two integer values a and b, the result  $a + b$  leaves both a and b unchanged).

What would you prefer – A separate class or a method in the HugelInteger objects? **Justify your preference...**

# Adding 2 HugeInteger objects



No reason to make one of the two objects we are adding more important than the other (would be different with something that would implement a kind of += operation).

A class method makes sense here.



- Give the pseudo code for the method

## Pseudo-code

```
HugeInteger result = new HugeInteger();  
                        // Default constructor needed for this
```

```
set min to the minimum size of the lists in h1 and h2  
set max to the maximum size of the lists in h1 and h2  
int sum;  
int val;  
int carry = 0;  
for (int i = 0; i < min; i++) {  
    sum = h1.list.get(i) + h2.list.get(i) + carry;  
    carry = sum / 1000;  
    result.list.add(sum % 1000);  
}
```

```
for (int i = min; i < max; i++) {  
    if i >= h1.list.size() {  
        val = h2.list.get(i);  
    } else {  
        val = h1.list.get(i);  
    }  
    sum = carry + val;  
    carry = sum / 1000;  
    result.list.add(sum % 1000);  
}  
if (carry > 0) {  
    result.list.add(carry);  
}  
return result;
```

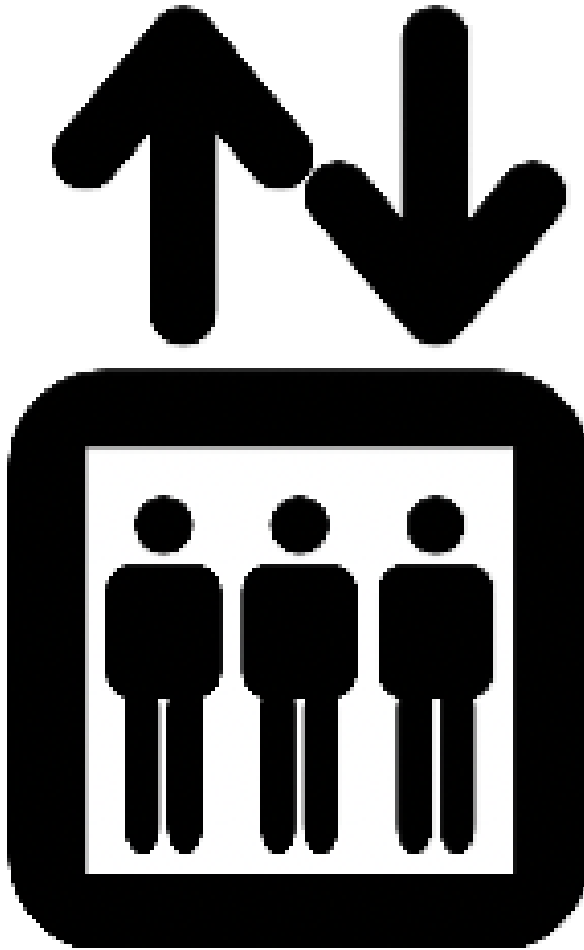
## Question – part 4

Use the previously defined method to write the pseudo code for a method that adds a long to a HugelInteger. You can assume it exists even if you didn't manage to write it.

# Adding a long to a HugelInteger

```
static HugelInteger add(HugelInteger h, long longVal) {  
    return add(h, new HugelInteger(longVal));  
}
```

Should also be overloaded with  
**add(long longVal, HugelInteger h)**  
to ensure commutativity (ie  $a * b = b * a$ ).



# Lift Example

## Problem Description

If you work for a lift manufacturer, you might have to code software for the operation of lifts. Optimizing the movements of lifts so as to minimize waiting times is not a small task. Knowing how many lifts are required in an office building depending on the number of people working in this building can also be challenging. One complicating factor is the need to model peak usage as people tend to use the lifts at the same time. Modelling and simulation can be used to do this, which takes us back to the roots of Object-Oriented programming.

# Implementation

We may think of creating a lift class. First of all, it must know which floors it serves. Then, it has a state: it may be stopped (for maintenance sometimes) or moving, up or down, and won't change direction before it has reached an "extreme stop". Messages are of two kinds: an "up" or "down" call from the outside (floor is known) and a "get me to floor" call when people press a button inside.



## Lift Class

- Floors served
- Direction
- Moving/Stopped
- Stop request

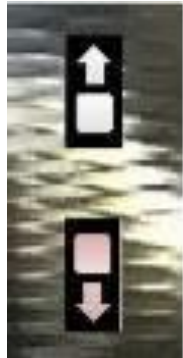




This model works when there is a single lift, but not when you have an "elevator lobby" served by several lifts. When you call a lift, you care very little about which lift will stop at your floor as long as it goes into the right direction.



# Controller Class

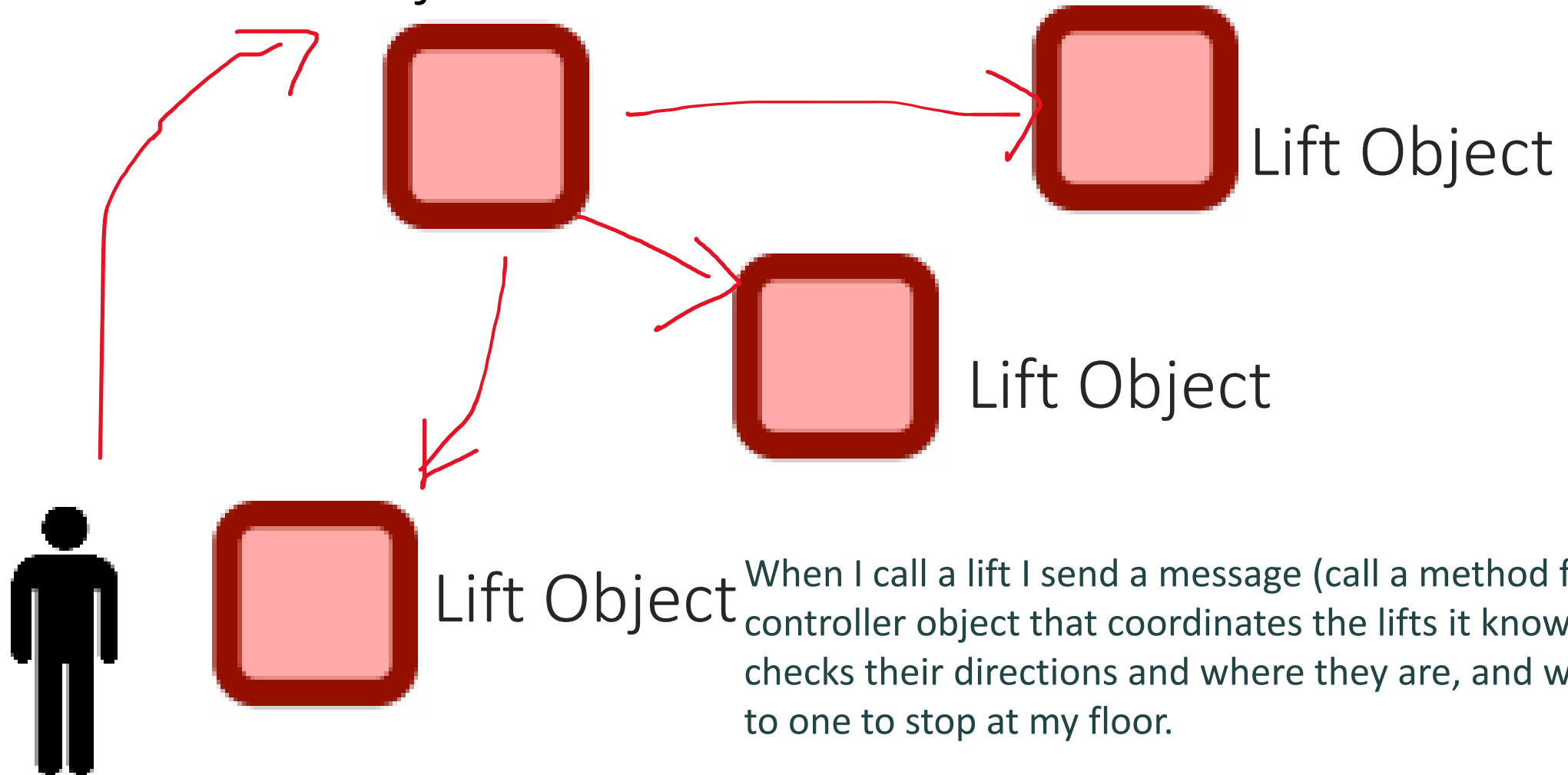


In that case we need a more abstract class, which I call "Controller class", for coordinating several lifts and taking call requests. Buttons inside the lift are still messages to a lift object that represents the lift you are in.

# Lift Class

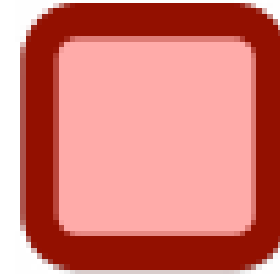
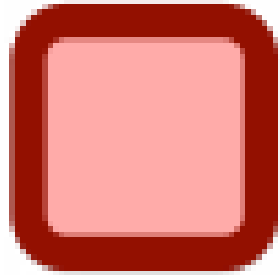


Controller Object

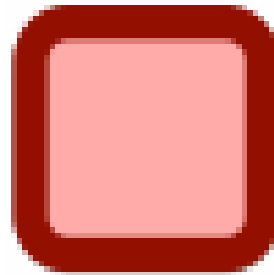


When I call a lift I send a message (call a method from) a controller object that coordinates the lifts it knows, checks their directions and where they are, and will ask to one to stop at my floor.

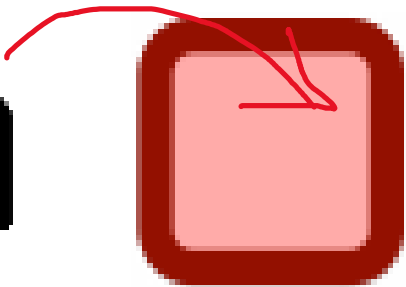
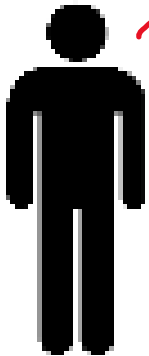
Controller Object



Lift Object



Lift Object



Lift Object

When a lift stops at my floor, I'll step into it and tell to that particular lift where I want to go (which the system doesn't know so far) by pressing a button inside the lift. And here is my object application.

# Creating an Object Oriented Application

- Identify necessary objects
- Define their role (what they do)
- Define the data they need
- Set up communications
- **Maximize consistency and minimize coupling**
- Trade-off between one object that does everything and objects that send too many messages

the hard part

# Question Set 1

1. What is the output of the following program?

```
public class ZeroDiv{  
  
    public static void main(String args[]){  
        int x = 0, y = 10;  
        try {  
            y /= x; }  
            System.out.print("/ by 0"); catch(Exception e) {  
                System.out.print("error");  
            }  
        }  
    }  
}
```

- a) 0
- b) Error
- c) Compilation fails
- d) An uncaught exception is thrown at runtime
- e) No output

# 1. What is the output of the following program?

Can't have a statement  
between try and catch

```
public class ZeroDiv{  
  
    public static void main(String args[]){  
        int x = 0, y = 10;  
        try {  
            y /= x; }  
            System.out.print("/ by 0"); catch(Exception e) {  
                System.out.print("error");  
            }  
        }  
    }  
}
```

- a) 0
- b) Error
- c) **Compilation fails**
- d) An uncaught exception is thrown at runtime
- e) No output

2. Which of the following statements about exception handling in Java are true?

- a) “throw” can be used to declare an exception in a method, and the exception will be thrown in this method
- b) “throws” is used to throw the exception objects
- c) “try” is used to detect if there is an exception in its block and if so it intercepts the exception and execute the code in the “catch” block
- d) No matter if there is an exception or not, the code in the “finally” block will be executed
- e) You cannot throw an exception in a “try” block



2. Which of the following statements about exception handling in Java are true?

- a) “throw” can be used to declare an exception in a method, and the exception will be thrown in this method
- b) “throws” is used to throw the exception objects
- c) “try” is used to detect if there is an exception in its block and if so it intercepts the exception and execute the code in the “catch” block
- d) No matter if there is an exception or not, the code in the “finally” block will be executed
- e) You cannot throw an exception in a “try” block

### 3. What is the output of the following code?

- a) Compilation error
- b) eE
- c) Ee
- d) eE1eE3eE5
- e) Ee1Ee3Ee5

```
public class Test {  
  
    private static void test(int[] arr) {  
        for (int i = 0; i < arr.length; i++) {  
            try {  
                if (arr[i] % 2 == 0) {  
                    throw new NullPointerException();  
                } else {  
                    System.out.print(i); }  
            } finally {  
                System.out.print("e"); }  
            }  
        }  
  
        public static void main(String[] args) {  
            try {  
                test(new int[] {0, 1, 2, 3, 4, 5});  
            } catch (Exception e) {  
                System.out.print("E");  
            }  
        }  
    }  
}
```

### 3. What is the output of the following code?

- a) Compilation error
- b) eE
- c) Ee
- d) eE1eE3eE5
- e) Ee1Ee3Ee5

```
public class Test {  
  
    private static void test(int[] arr) {  
        for (int i = 0; i < arr.length; i++) {  
            try {  
                if (arr[i] % 2 == 0) {  
                    throw new NullPointerException();  
                } else {  
                    System.out.print(i); }  
            } finally {  
                System.out.print("e"); }  
            }  
        }  
  
        public static void main(String[] args) {  
            try {  
                test(new int[] {0, 1, 2, 3, 4, 5});  
            } catch (Exception e) {  
                System.out.print("E");  
            }  
        }  
    }  
}
```

An exception is a type of object which can be thrown by any other class/method. Here method test throws the null pointer exception, the finally block is always executed in a try-catch block so “e” is printed, and then in main the null pointer exception is caught resulting in printing “E”

#### 4. What is the output of the following code?

```
public class Test {  
  
    public static void main(String[] args) {  
        System.out.println("return value of getValue(): "  
                           + getValue());  
    }  
  
    public static int getValue() {  
        try {  
            return 0;  
        } finally {  
            return 1;  
        }  
    }  
}
```

#### 4. What is the output of the following code?

```
public class Test {  
  
    public static void main(String[] args) {  
        System.out.println("return value of getValue(): "  
                           + getValue());  
    }  
  
    public static int getValue() {  
        try {  
            return 0;  
        } finally {  
            return 1;  
        }  
    }  
}
```

return value of getValue(): 1

5. Which is the correct statement about exception handling in Java?

- a) If you have defined a possible exception in a method with “throw” you definitely have this exception when you use the method
- b) If there is no exception thrown in the “try” block then the code in the “finally” block won’t be executed
- c) If your program throws an exception then there must be an error in your program – you need to debug and fix the error
- d) An unchecked exception in Java derives from RuntimeException or its children

5. Which is the correct statement about exception handling in Java?

- a) If you have defined a possible exception in a method with “throw” you definitely have this exception when you use the method
- b) If there is no exception thrown in the “try” block then the code in the “finally” block won’t be executed
- c) If your program throws an exception then there must be an error in your program – you need to debug and fix the error
- d) An unchecked exception in Java derives from RuntimeException or its children

6. You don't need a "finally" block to release resources if you wrote your "try" block as follows:

```
try (// Acquire resources here) {  
    ...  
} catch ... {  
}
```

- a) True
- b) False



6. You don't need a "finally" block to release resources if you wrote your "try" block as follows:

```
try (// Acquire resources here) {  
    ...  
} catch ... {  
}
```

a) True

b) False

7. Which of the following is incorrect:

- a) A “try” block cannot be omitted
- b) Multiple “catch” blocks can be used
- c) A “finally” block can be omitted
- d) A “finally” block can be used without any “try” or “catch” block

7. Which of the following is incorrect:

- a) A “try” block cannot be omitted
- b) Multiple “catch” blocks can be used
- c) A “finally” block can be omitted
- d) A “finally” block can be used without any “try” or “catch” block

8. Suppose your program reads a value entered by the user. How should you create a custom exception that is thrown if the input is greater than 10?

- a) `if (i > 10) throw new Exception("something's wrong!");`
- b) `if (i > 10) throw Exceptione("something's wrong!");`
- c) `if (i > 10) throw new Exceptione("something's wrong!");`
- d) `if (i > 10) throw Exception("something's wrong!");`

8. Suppose your program reads a value entered by the user. How should you create a custom exception that is thrown if the input is greater than 10?

- a) `if (i > 10) throw new Exception("something's wrong!");`
- b) `if (i > 10) throw Exceptione("something's wrong!");`
- c) `if (i > 10) throw new Exceptione("something's wrong!");`
- d) `if (i > 10) throw Exception("something's wrong!");`

9. In the program below where will the references to the variables a, b, and c, be stored?

- a) Heap, heap, heap
- b) Heap, stack, heap
- c) Heap, stack, stack
- d) Heap, heap, stack

```
class A {  
    private String a = "aa";  
  
    public boolean methodB() {  
        String b = "bb";  
        final String c = "cc";  
    }  
}
```

9. In the program below where will the references to the variables a, b, and c, be stored?

- a) Heap, heap, heap
- b) Heap, stack, heap
- c) **Heap, stack, stack**
- d) Heap, heap, stack

```
class A {  
    private String a = "aa";  
  
    public boolean methodB() {  
        String b = "bb";  
        final String c = "cc";  
    }  
}
```

"aa", "bb" and "cc" are all in the heap but we are talking about references to these values.

10. What characterizes the Set interface?

- a) A set is a collection of elements which contains elements along with their key
- b) A Set is a collection of elements which contains hashcode of elements
- c) A set is a collection of elements which cannot contain duplicates
- d) A set is a collection that is always ordered



## 10. What characterizes the Set interface?

- a) A set is a collection of elements which contains elements along with their key
- b) A Set is a collection of elements which contains hashcode of elements
- c) A set is a collection of elements which cannot contain duplicates
- d) A set is a collection that is always ordered

11. What is the parent class of Error and Exception classes?

- a) Throwable
- b) Catchable
- c) MainError
- d) MainException

11. What is the parent class of Error and Exception classes?

- a) Throwable
- b) Catchable
- c) MainError
- d) MainException

12. Which arithmetic operations can (together or alone) result in the throwing of ArithmeticException?

- a) /, %
- b) \*, +
- c) !, -
- d) <<, >>

12. Which arithmetic operations can (together or alone) result in the throwing of ArithmeticException?

a) **/, %**

b) \*, +

c) !, -

d) <<, >>

13. The following are descriptions about List interface, Set interface and Map interface, which is false?

- a) They all inherit from the Collection interface
- b) List is an ordered interface, so we can control precisely where each element is inserted when using the interface
- c) Set is a collection that does not contain duplicate elements
- d) Map provides a mapping from key to value and Map cannot contain the same key several times, each key can only map a value

13. The following are descriptions about List interface, Set interface and Map interface, which is false?

- a) They all inherit from the Collection interface
- b) List is an ordered interface, so we can control precisely where each element is inserted when using the interface
- c) Set is a collection that does not contain duplicate elements
- d) Map provides a mapping from key to value and Map cannot contain the same key several times, each key can only map a value

14. Of the following statements about the Collections class, which is **false**?

- a) Both ArrayList and LinkedList implement the List interface
- b) Access to elements of an ArrayList is faster than elements of a LinkedList
- c) When adding and removing elements, ArrayList performs better than LinkedList
- d) HashMap implements the Map interface, which allows any type of key and value object and allows null to be used as a key or value



14. Of the following statements about the Collections class, which is **false**?

- a) Both ArrayList and LinkedList implement the List interface
- b) Access to elements of an ArrayList is faster than elements of a LinkedList
- c) When adding and removing elements, ArrayList performs better than LinkedList
- d) HashMap implements the Map interface, which allows any type of key and value object and allows null to be used as a key or value

Performs worse because potentially will need to shift elements to make a slot

15. Which of the following interfaces are directly inherited from the Collection interface (multiple answers)?

- a) List
- b) Map
- c) Set
- d) Iterator

15. Which of the following interfaces are directly inherited from the Collection interface (multiple answers)?

- a) List
- b) Map
- c) Set
- d) Iterator

16. The following is statement is about Collection and Collections, which is **true**?

- a) Collection is a class under java.util which contains various static methods for collection operations
- b) Collection is a class under java.util which is the parent interface of various collection structures
- c) Collections is a class under java.util which is the parent interface for the various collection structures
- d) Collections is a class under java.util which contains various static methods for collection operations

16. The following is statement is about Collection and Collections, which is **true**?

- a) Collection is a class under java.util which contains various static methods for collection operations
- b) Collection is a class under java.util which is the parent interface of various collection structures
- c) Collections is a class under java.util which is the parent interface for the various collection structures
- d) Collections is a class under java.util which contains various static methods for collection operations

17. What will be printed out on running the following program

- a) 2,2
- b) 2,3
- c) 3,2
- d) 3,3

```
public class Test{  
    public static void main(String [] args){  
        List list=new ArrayList();  
        list.add("a");  
        list.add("b");  
        list.add("a");  
  
        Set set=new HashSet();  
        set.add("a");  
        set.add("b");  
        set.add("a");  
  
        System.out.println(list.size()+" "+set.size());  
    }  
}
```

17. What will be printed out on running the following program

a) 2,2

b) 2,3

c) 3,2

d) 3,3

```
public class Test{
    public static void main(String [] args){
        List list=new ArrayList();
        list.add("a");
        list.add("b");
        list.add("a");

        Set set=new HashSet();
        set.add("a");
        set.add("b");
        set.add("a");

        System.out.println(list.size()+" "+set.size());
    }
}
```

18. After the code below is executed what are the elements in NumberList (if any)?

a) 2,4,1,3,5

b) 2,1,3,5

c) 4,1,3,5

d) There will be an out of bounds exception

```
List<Integer> NumberList = new ArrayList<Integer>();
NumberList.add(2);
NumberList.add(4);
NumberList.add(1);
NumberList.add(3);
NumberList.add(5);
for(int i = 0; i < NumberList.size(); ++i) {
    int v = NumberList.get(i);
    if(v%2 == 0) {
        NumberList.remove(v);
    }
}
System.out.println(NumberList);
```



18. After the code below is executed what are the elements in NumberList (if any)?

a) 2,4,1,3,5

b) 2,1,3,5

c) 4,1,3,5

d) There will be an out of bounds exception

```
List<Integer> NumberList = new ArrayList<Integer>();
NumberList.add(2);
NumberList.add(4);
NumberList.add(1);
NumberList.add(3);
NumberList.add(5);
for(int i = 0; i < NumberList.size(); ++i) {
    int v = NumberList.get(i);
    if(v%2 == 0) {
        NumberList.remove(v);
    }
}
System.out.println(NumberList);
```

19. A local variable in a method can be public.

- a) True
- b) False

19. A local variable in a method can be public.

a) True

b) False

20. What will happen when the following code is executed?

```
class MyError extends Error{ }

public class TestError {
    public static void main(String args[]) {
        try {
            test();
        } catch(Error ie) {
            System.out.println("Error caught");
        }
    }

    static void test() throws Error {
        throw new MyError();
    }
}
```

- a) Runtime error test() method does not throw an error type instance
- b) Compile time error Cannot catch Error type objects
- c) Compile time error Error class cannot be extended
- d) Prints "Error caught"

20. What will happen when the following code is executed?

```
class MyError extends Error{ }

public class TestError {
    public static void main(String args[]) {
        try {
            test();
        } catch(Error ie) {
            System.out.println("Error caught");
        }
    }

    static void test() throws Error {
        throw new MyError();
    }
}
```

- a) Runtime error test() method does not throw an error type instance
- b) Compile time error cannot catch Error type objects
- c) Compile time error Error class cannot be extended
- d) Prints "Error caught"

21. Annotations about annotations are called:

- a) Depreciations
- b) Metadata
- c) Meta-annotations
- d) There is no such thing

21. Annotations about annotations are called:

- a) Depreciations
- b) Metadata
- c) **Meta-annotations**
- d) There is no such thing

22. If a method is tagged with `@Deprecated` then:

- a) Nothing special happens it is just information for javac
- b) Javac will issue a warning but still generate the .class file
- c) Javac ends in an error



22. If a method is tagged with @Deprecated then:

- a) Nothing special happens it is just information for javac
- b) Javac will issue a warning but still generate the .class file
- c) Javac ends in an error

It will "throw a warning if someone uses this method" but nothing happens when compiling the class where the annotation is added.

23. If a Java interface defines two or more methods you cannot use lambda expressions with it.

- True
- False

23. If a Java interface defines two or more methods you cannot use lambda expressions with it.

- True
- False

24. In a Graphical User Interface, you can only store a container in a different type of container (for instance you can only add a vertical box to a grid or a horizontal box but not another vertical box).

- a) True
- b) False

24. In a Graphical User Interface, you can only store a container in a different type of container (for instance you can only add a vertical box to a grid or a horizontal box but not another vertical box).

a) True

b) False

25. If you are importing a .css file (style sheet) in a JavaFx application and the file cannot be found then:

- a) If you haven't inserted the CSS file loading into a try .. catch .. block the application crashes
- b) There is no exception thrown at the JavaFx application level. The application doesn't crash without a try .. catch .. block but it exits
- c) There is no exception thrown at the JavaFx application level. The application writes a warning to the console and continues with default styling.

25. If you are importing a .css file (style sheet) in a JavaFx application and the file cannot be found then:

- a) If you haven't inserted the CSS file loading into a try .. catch .. block the application crashes
- b) There is no exception thrown at the JavaFx application level. The application doesn't crash without a try .. catch .. block but it exits
- c) There is no exception thrown at the JavaFx application level. The application writes a warning to the console and continues with default styling.

26. A JavaFx application must have an `init()`, `start()` and `stop()` method.

- a) True
- b) False



26. A JavaFx application must have an init(), start() and stop() method.

a) True

b) False

Only start() is abstract in the Application class and has to be rewritten.

# Question Set 2

1. What would the following code display?

```
int[] a = {1,2,3,4,5,6};  
int i = a.length - 1;  
while(i>=0){  
    System.out.print(a[i]);  
    i--;  
}
```

- a) 123456
- b) Exception at runtime
- c) 654321
- d) Nothing
- e) 65432
- f) 12345

1. What would the following code display?

```
int[] a = {1,2,3,4,5,6};  
int i = a.length - 1;  
while(i>=0){  
    System.out.print(a[i]);  
    i--;  
}
```

- a) 123456
- b) Exception at runtime
- c) 654321
- d) Nothing
- e) 65432
- f) 12345

Pretty obvious once you have understood that it's looping in decreasing order and when you have the boundaries right.

2. A and E are classes, B and D are interfaces. Which of the following statements are correct?

- a) interface F implements B,D { }
- b) interface F implements D { }
- c) interface F extends D { }
- d) interface F extends E { }
- e) interface F extends B, D { }

2. A and E are classes, B and D are interfaces. Which of the following statements are correct?

- a) interface F implements B,D { }
- b) interface F implements D { }
- c) interface F extends D { }
- d) interface F extends E { }
- e) interface F extends B, D { }

"To implement" means to provide the code for a method specified in an interface. An interface can only extend, and extend **one** other interface (no multiple inheritance in Java)

### 3. What is inheritance?

- a) It is the process where one object acquires properties of another
- b) Inheritance is the ability of an object to take on many forms
- c) Inheritance is a technique to define different methods of the same type
- d) None of the above

### 3. What is inheritance?

- a) It is the process where one object acquires properties of another
- b) Inheritance is the ability of an object to take on many forms
- c) Inheritance is a technique to define different methods of the same type
- d) None of the above

b refers to "polymorphism" and c (although it says nothing about method names) \*might\* refer to overloading.



#### 4. What is polymorphism

- a) Polymorphism is a technique to define different objects of the same type
- b) Polymorphism is the ability of an object to take on many forms
- c) Polymorphism is a technique to define different methods of the same type
- d) None of the above

## 4. What is polymorphism

- a) Polymorphism is a technique to define different objects of the same type
- b) Polymorphism is the ability of an object to take on many forms
- c) Polymorphism is a technique to define different methods of the same type
- d) None of the above

Poly = several, morph = shape (in Greek)

## 5. What are instance variables?

- a) Instance variables are static variables within a class but outside any method
- b) Instance variables are variables defined inside methods, constructors or blocks
- c) Instance variables are variables within a class but outside any method
- d) None of the above

## 5. What are instance variables?

- a) Instance variables are static variables within a class but outside any method
- b) Instance variables are variables defined inside methods, constructors or blocks
- c) Instance variables are variables within a class but outside any method
- d) None of the above

*static* variables may be class variables (outside a method) or variables that you only see within a method but retain their values across calls (they aren't on the stack); b refers to *local* variables, allocated on the stack.

6. When we refer to a “member of a class” we mean:

- a) An attribute
- b) A method
- c) An attribute or method
- d) A sub-class

6. When we refer to a “member of a class” we mean:

- a) An attribute
- b) A method
- c) An attribute or method
- d) A sub-class

"member" is a generic term for whatever belongs to a larger group (group member, family member ...).  
Anything inside a class is a "member".

7. What would be displayed after the following code?

```
int[] nums = {1,2,3,4,5,6};  
System.out.println((nums[1] + nums[3]));
```

- a) 6
- b) 2+4
- c) 1+3
- d) 4

7. What would be displayed after the following code?

```
int[] nums = {1,2,3,4,5,6};  
System.out.println((nums[1] + nums[3]));
```

- a) 6
- b) 2+4
- c) 1+3
- d) 4

Easy...



8. If classes Student, Staff, and Faculty extend the class Person, which of the following make sense?

- a) `Faculty[] faculties = {new Person(), new Staff(), new Student()};`
- b) `Staff[] staff={new Person(), new Faculty(), new Student()};`
- c) `Person[] persons={new Faculty(), new Staff(), new Student()};`

8. If classes Student, Staff, and Faculty extend the class Person, which of the following make sense?

- a) `Faculty[] faculties = {new Person(), new Staff(), new Student()};`
- b) `Staff[] staff={new Person(), new Faculty(), new Student()};`
- c) `Person[] persons={new Faculty(), new Staff(), new Student()};`

A reference to the parent can be assigned a reference to a child, not the reverse. Every Faculty, Student and Staff is a Person.

9. What is the output of the following program?

- a) 5
- b) 18
- c) 3
- d) Compile error
- e) Runtime error

```
class Recursion {
    int func(int n) {
        int result;
        if (n == 0) {
            result = 3;
        } else {
            result = func(n - 1);
        }
        return result;
    }
}

public class Prog {
    public static void main(String args[]) {
        Recursion obj = new Recursion();
        System.out.print(obj.func(5));
    }
}
```

9. What is the output of the following program?

- a) 5
- b) 18
- c) 3
- d) Compile error
- e) Runtime error

```
class Recursion {  
    int func(int n) {  
        int result;  
        if (n == 0) {  
            result = 3;  
        } else {  
            result = func(n - 1);  
        }  
        return result;  
    }  
}
```

The function returns the value for a smaller n until it hits zero and returns 3, so it can only return 3.

```
public class Prog {  
    public static void main(String args[]) {  
        Recursion obj = new Recursion();  
        System.out.print(obj.func(5));  
    }  
}
```

10. What is the output of the following program?

- a) 0 0 0
- b) 0 2 1
- c) 0 1 0
- d) Compile error

```
class Recursion {
    int func(int n) {
        int result = 0;
        if (n < 10) {
            if (n == 8) {
                result = 1;
            }
        } else {
            result = func(n % 10) + func(n / 10);
        }
        return result;
    }
}

public class Prog {
    public static void main(String args[]) {
        Recursion obj = new Recursion();
        System.out.print(obj.func(123) + " ");
        System.out.print(obj.func(8328) + " ");
        System.out.println(obj.func(8325));
    }
}
```

10. What is the output of the following program?

a) 0 0 0

b) 0 2 1

The function actually counts how many '8' there are in the number.

c) 0 1 0

d) Compile error

```
class Recursion {
    int func(int n) {
        int result = 0;
        if (n < 10) {
            if (n == 8) {
                result = 1;
            }
        } else {
            result = func(n % 10) + func(n / 10);
        }
        return result;
    }
}

public class Prog {
    public static void main(String args[]) {
        Recursion obj = new Recursion();
        System.out.print(obj.func(123) + " ");
        System.out.print(obj.func(8328) + " ");
        System.out.println(obj.func(8325));
    }
}
```

11. Which of the following is a valid annotation definition?

- a) `public @annotation MyAnnotation{ }`
- b) `private @interface MyAnnotation{ }`
- c) `public @interface MyAnnotation{ }`
- d) `public @MyAnnotation{ }`

11. Which of the following is a valid annotation definition?

- a) `public @annotation MyAnnotation{ }`
- b) `private @interface MyAnnotation{ }`
- c) `public @interface MyAnnotation{ }`
- d) `public @MyAnnotation{ }`

An annotation is a special interface, and there is no reason to make it private.



12. Which of the following belongs to meta-annotations?

- a) `@Override`
- b) `@Retention`
- c) `@Deprecated`
- d) `@SuppressWarnings()`

## 12. Which of the following belongs to meta-annotations?

- a) `@Override`
- b) `@Retention`
- c) `@Deprecated`
- d) `@SuppressWarnings()`

A meta-annotation is an annotation about annotation. The other annotations refer to the annotated code.

FYI: The `@Retention` annotation indicates how long annotations with the annotated type are to be retained.

Can be one of:

- **CLASS:** annotations are to be recorded in the class file by the compiler but need not be retained by the VM at run time.
- **RUNTIME:** Annotations are to be recorded in the class file by the compiler and retained by the VM at run time, so they may be read reflectively.
- **SOURCE:** Annotations are to be discarded by the compiler.

13. Which of the following are valid retention policy types in Java (multiple answers)?

- a) SOURCE
- b) CLASS
- c) RUNTIME
- d) CODE
- e) TOOLS

13. Which of the following are valid retention policy types in Java (multiple answers)?

- a) SOURCE
- b) CLASS
- c) RUNTIME
- d) CODE
- e) TOOLS

CODE and TOOLS don't exist.

## 14. What will the following program output?

- a) Prints HelloAnnotation 10
- b) Good Evening Prints 10
- c) Prints 10
- d) Some other output
- e) Output cannot be determined

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@interface MyAnnotation {
    int value();
}

class Hello {
    @MyAnnotation(value = 10)
    public void goodEvening() {
        System.out.print("Good Evening");
    }
}

public class HelloAnnotation {
    public static void main(String args[]) throws Exception {
        Hello h = new Hello();
        Method m = h.getClass().getMethod("goodEvening");
        MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
        System.out.println("Prints " + myAnn.value());
    }
}
```

## 14. What will the following program output?

- a) Prints HelloAnnotation 10
- b) Good Evening Prints 10
- c) Prints 10
- d) Some other output
- e) Output cannot be determined

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.RUNTIME)
@interface MyAnnotation {
    int value();
}

class Hello {
    @MyAnnotation(value = 10)
    public void goodEvening() {
        System.out.print("Good Evening");
    }
}

public class HelloAnnotation {
    public static void main(String args[]) throws Exception {
        Hello h = new Hello();
        Method m = h.getClass().getMethod("goodEvening");
        MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
        System.out.println("Prints " + myAnn.value());
    }
}
```

The method goodEvening() isn't called, we only retrieve the annotation.

15. What will be the output of the following program?

- a) Prints HelloAnnotation 10
- b) Compilation Error
- c) Runtime Exception

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.CLASS)
@interface MyAnnotation {
    int value();
}

class Hello {
    @MyAnnotation(value = 10)
    public void goodEvening() {
        System.out.print("Good Evening");
    }
}

public class HelloAnnotation { when the program runs.
    public static void main(String args[]) throws Exception {
        Hello h = new Hello();
        Method m = h.getClass().getMethod("goodEvening");
        MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
        System.out.println("Prints " + myAnn.value());
    }
}
```

## 15. What will be the output of the following program?

- a) Prints HelloAnnotation 10
- b) Compilation Error
- c) **Runtime Exception**

As the retention isn't RUNTIME,  
trying to get the annotation  
returns a null reference when the  
program runs

```
import java.lang.annotation.*;
import java.lang.reflect.*;

@Retention(RetentionPolicy.CLASS)
@interface MyAnnotation {
    int value();
}

class Hello {
    @MyAnnotation(value = 10)
    public void goodEvening() {
        System.out.print("Good Evening");
    }
}

public class HelloAnnotation { when the program runs.
    public static void main(String args[]) throws Exception {
        Hello h = new Hello();
        Method m = h.getClass().getMethod("goodEvening");
        MyAnnotation myAnn = m.getAnnotation(MyAnnotation.class);
        System.out.println("Prints " + myAnn.value());
    }
}
```



16. What is the output of the following program?

- a) false true
- b) true false
- c) true false
- d) false false

```
interface Int{}
class Simple implements Int{}
class SimpleTest {
    public static void main(String[] args) {
        try {
            Class c=Class.forName("Simple");
            System.out.print(c.isInterface() + " ");
            c=Class.forName("Int");
            System.out.println(c.isInterface());
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

16. What is the output of the following program?

- a) false true
- b) true false
- c) true false
- d) false false

An interface kind of  
extends class (lightweight  
abstract class)

```
interface Int{}
class Simple implements Int{}
class SimpleTest {
    public static void main(String[] args) {
        try {
            Class c=Class.forName("Simple");
            System.out.print(c.isInterface() + " ");
            c=Class.forName("Int");
            System.out.println(c.isInterface());
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}
```

17. Which of the following can obtain the location of file `Reflection.class`

Public class `Reflection { }`

- a) `Reflection.class.getClassLoader().getResource("Reflection.class")`
- b) `Reflection.getClass().getResource("Reflection.class")`
- c) `Reflection.getClass().getClassLoader().getResource("Reflection.class")`
- d) `Reflection.class.getClassLoader("Reflection.class")`

17. Which of the following can obtain the location of file Reflection.class

Public class Reflection { }

- a) Reflection.class.getClassLoader().getResource("Reflection.class")
- b) Reflection.getClass().getResource("Reflection.class")
- c) Reflection.getClass().getClassLoader().getResource("Reflection.class")
- d) Reflection.class.getClassLoader("Reflection.class")

getClass() applies to an object, .class to a class. File "Reflection.class" is a resource for the JVM.

18. Consider the following program:

Which of the following is true?

- a) It generates one warning (call of the deprecated constructor) when compiling, nothing when running
- b) It generates two warnings (definition of the deprecated constructor and call of the deprecated constructor) when compiling, nothing when running
- c) No warning when compiling but it generates two warnings when running
- d) No warning when compiling, exception when running
- e) One warning when compiling, exception when running

```
class MyClass {  
    private float val;  
    @Deprecated  
    MyClass(int n) {  
        val = n;  
    }  
    MyClass(float f) {  
        val = f;  
    }  
}  
  
public class TestDeprecated {  
    public static void main(String[] args) {  
        int val = 3;  
        MyClass obj = new MyClass(3);  
    }  
}
```

18. Consider the following program:

Which of the following is true?

- a) It generates one warning (call of the deprecated constructor) when compiling, nothing when running
- b) It generates two warnings (definition of the deprecated constructor and call of the deprecated constructor) when compiling, nothing when running
- c) No warning when compiling but it generates two warnings when running
- d) No warning when compiling, exception when running
- e) One warning when compiling, exception when running

```
class MyClass {  
    private float val;  
    @Deprecated  
    MyClass(int n) {  
        val = n;  
    }  
    MyClass(float f) {  
        val = f;  
    }  
}  
  
public class TestDeprecated {  
    public static void main(String[] args) {  
        int val = 3;  
        MyClass obj = new MyClass(3);  
    }  
}
```

Only developers USING the deprecated function with javac are warned.

19. What is the main purpose of JDBC?

- a) Create a connection to the database.
- b) Send SQL statements to the database.
- c) Processing data and query results
- d) All of the above

19. What is the main purpose of JDBC?

- a) Create a connection to the database.
- b) Send SQL statements to the database.
- c) Processing data and query results
- d) All of the above

Easy...



20. Which of the following steps are performed when accessing a database via JDBC?

- a) Loading the JDBC driver.
- b) Setting up the connection.
- c) Executing queries or applying changes to the database
- d) Close the connection

20. Which of the following steps are performed when accessing a database via JDBC?

- a) Loading the JDBC driver.
- b) Setting up the connection.
- c) Executing queries or applying changes to the database
- d) Close the connection

Note: although 99.9% of JDBC applications load the driver, it's also possible to import it (but then the .jar must be in your classpath when you compile)

21. Which package allows Java to access a database?

- a) java.sql
- b) java.db
- c) java.dbms
- d) java.jdbc
- e) java.lang
- f) java.util

21. Which package allows Java to access a database?

- a) `java.sql`
- b) `java.db`
- c) `java.dbms`
- d) `java.jdbc`
- e) `java.lang`
- f) `java.util`

b, c, d don't exist. e is the default package. f contains a lot of useful stuff, but not JDBC.

22. In general you prefer adding a TableView to a:

- a) BorderPane
- b) StackPane
- c) ScrollPane
- d) SplitPane
- e) TabPane
- f) TitledPane

22. In general you prefer adding a TableView to a:

- a) BorderPane
- b) StackPane
- c) ScrollPane
- d) SplitPane
- e) TabPane
- f) TitledPane

A TableView is used to display data retrieved from a data source (often a database, but it could as well be the result of streaming an in-memory collection, or data retrieved from the network). You rarely know how many rows you'll display, and the wisest choice is a ScrollPane.

## 23. FXCollections:

- a) Is a package that contains classes for collections that you should use as backend to some JavaFX widgets.
- b) Is a class that contains wrapper methods for regular collections so that you can use them as backend to some JavaFX widgets.
- c) Is a class that only contains the ObservableList inner class
- d) Is an enum that is used for defining the collections that can be used with JavaFx widgets (for instance a TreeSet cannot be used, and doesn't appear in the enum)

## 23. FXCollections:

- a) Is a package that contains classes for collections that you should use as backend to some JavaFX widgets.
- b) Is a class that contains wrapper methods for regular collections so that you can use them as backend to some JavaFX widgets.
- c) Is a class that only contains the ObservableList inner class
- d) Is an enum that is used for defining the collections that can be used with JavaFx widgets (for instance a TreeSet cannot be used, and doesn't appear in the enum)

In Java, plural is often used to name classes that contain static methods (Arrays, Collections, ...)



24. “static” key word in java indicates:

- a) Something that you cannot change
- b) Something that is attached to the class, not to a particular object instance
- c) Something that cannot throw exceptions
- d) Something that cannot be overridden (methods) or extended

24. “static” key word in java indicates:

- a) Something that you cannot change
- b) Something that is attached to the class, not to a particular object instance
- c) Something that cannot throw exceptions
- d) Something that cannot be overridden (methods) or extended

Something that you cannot change (variable) or cannot be overridden (method) or extended (class) is preceded by *final*.

25. The protocol that takes a message from a network to another another network is:

- a) FTP
- b) TCP
- c) IP
- d) HTTP

25. The protocol that takes a message from a network to another another network is:

a) FTP

IP = Internet Protocol.

*Internet* means to net(work)s what *international* means to nations.

b) TCP

FTP = File Transfer Protocol (more like HTTP, but dedicated to exchanging files)

c) IP

TCP = Transmission Control Protocol, computer-to-computer (uses IP underneath)

d) HTTP

HTTP = Hyper Text Transfer Protocol, you know this one.

26. Which of these is **not** a build tool?

- a) Ant
- b) Junit
- c) Make
- d) Maven

26. Which of these is **not** a build tool?

a) Ant

b) Junit Testing tool.

c) Make

d) Maven

27. The time taken by a message to travel between computers is called:

- a) Bandwidth
- b) Routing
- c) Latency
- d) Delay

27. The time taken by a message to travel between computers is called:

- a) Bandwidth
- b) Routing
- c) Latency
- d) Delay

Bandwidth refers to how much data can be sent at the same time, not to speed. Routing is finding a circuit (route) to go to computer to computer; it's about traffic redirection.  
Delay is just a general term.



28. You need to have the .jar of the JDBC driver you are using when:

- a) Only when you compile the program
- b) Only when you run the program
- c) Both when you compile and run the program

28. You need to have the .jar of the JDBC driver you are using when:

- a) Only when you compile the program
- b) Only when you run the program
- c) Both when you compile and run the program

b is the good answer for 99.9% of the cases, when you load the driver dynamically. Now, you can also import it, but in that case the good answer would be c. And if you are using JavaDB (Derby), as it's part of the standard JDK/JRE you don't need to specify anything special.

A quiz isn't the place for fine-point distinctions. Always pick the most likely.

29. If you serialize an object that contains references, when you reload it the referenced objects will go at exactly the same place in memory.

- a) True
- b) False

29. If you serialize an object that contains references, when you reload it the referenced objects will go at exactly the same place in memory.

a) True

b) False

Memory location is the business of the operating system, not yours. It can put things wherever it likes, as long as it returns references to you.

30. To make an object serializable you:

- a) You don't need to do anything – all objects are serializable by default
- b) You only need to say that it implements the Serializable interface
- c) You need to say that it implements the Serializable interface and implement the two methods `writeObject()` and `readObject()` defined in the interface

30. To make an object serializable you:

- a) You don't need to do anything – all objects are serializable by default
- b) You only need to say that it implements the Serializable interface
- c) You need to say that it implements the Serializable interface and implement the two methods `writeObject()` and `readObject()` defined in the interface

It's not done by default because it adds overhead (more code) that not everybody needs.