lab 2: GCD

find GCD of 12 and 24

\[ 24 = 2 \cdot 12 + 0 \rightarrow 12 = 1 \]

try 12, 13:

\[ 13 = 1 \cdot 12 + 1 \]
\[ 12 = 12 \cdot 1 + 0 \rightarrow 1 = 1 \]

try 12, 24 again

\[ 12 = 0 \cdot 24 + 12 \]
\[ 24 = 2 \cdot 12 + 0 \rightarrow 12 = 1 \]
Abstract class

An abstract method is simply a method heading, ending with ":"

abstract type method_name(...param...)

used to define implementation of a method to a subclass.

An abstract class is one that contains one or more abstract methods.
General form

abstract class MyClass {

    // fields
    // methods

    // at least one abstract method
    abstract type myMethod();
}

- any subclass of MyClass (that is not itself abstract) is obligated to define myMethod().
- abstract classes cannot be instantiated, i.e. have no constructor.
**Example:** Abstract Counter

```
Abstract Counter
   /    \
One Counter  Two Counter
```

**Example:** Pair-Counter

Counts # of 2-element subsets of a set with clickCount many elements

<table>
<thead>
<tr>
<th>clickCount</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>15</td>
</tr>
</tbody>
</table>
Ex. \( \binom{1}{1}, 2, 3 \)

2-subsets: \( \{1, 2\}, \{1, 3\}, \{2, 3\} \)

Ex. \( \binom{1}{1}, 2, 3 \)

2-subsets: \( \{1, 2\}, \{1, 3\}, \{1, 4\} \)

\( \{2, 3\}, \{2, 4\} \)

\( \{3, 4\} \)

Ex. \( \binom{1}{1}, 2, 3, 4, 5 \)

2-subsets: \( \{1, 2\}, \{1, 3\}, \{1, 4\}, \{1, 5\} \)

\( \{2, 3\}, \{2, 4\}, \{2, 5\} \)

\( \{3, 4\}, \{3, 5\} \)

\( \{4, 5\} \)

_read: \"2-choose-2\"

\( \binom{5}{2} \) or \( \binom{5}{2} \)

or \( C_2 \)
Interfaces

An interface in Java is a class-like structure encapsulating only

- abstract methods
- constants (no variables)

all implementation details are deferred.

Rudex

- Interface methods are implicitly public, therefore public is usually left off

- Same for abstract

- A class that inherits from an interface is said to implement that interface.
General form:

```java
// myInterface.java
interface myInterface{
    // constants
    // abstract methods
}
```

```java
// myClass.java
class myClass implements myInterface{
    // definition of
    // abstract methods
    // other stuff
}
```
Note:

We can extend only one class, but can implement multiple interfaces.