Executing Java Programs

- `H:\...\ExtraProjects\uglab> java UGLab`
  - “java” starts the Java virtual machine.
  - The named class is loaded and execution is started.
  - Other classes are loaded as needed.
  - Only possible if class has been compiled.

- Problem: Execute what?
- If we try:
  - `H:\...\ExtraProjects\uglab> java UGLab`
    Exception in thread "main"
    java.lang.NoSuchMethodError: main

- Problem: how does the system know which method to execute?
The main method

• The answer: The java system always executes a method called main with a certain signature:

   public static void main(String[] args)
   {
    ...
   }

• For this to work, such a method must exist!
main method

- "main" must exist
- "main" must be public
- "main" must be static (class method)
- "main" must have a String array parameter
- Only "main" can be invoked
Example 1

```java
public static void main(String[] args) {
    UGLab lab = new UGLab();
    Student student1 = new Student("Rudi", 3);
    Student student2 = new Student("Sue", 7);
    lab.add(student1);
    lab.add(student2);
    lab.run(200);
}
```
Example 2

• Often it is simpler than this - the main method will
  – create an object
  – call the first method

• e.g.

  public static void main(String[] args) {
    Game game = new Game();
    game.play();
  }
What is String[] args?

- Q: What is the String[] args parameter for?
- A: For command line arguments
- These are arguments (parameters) you can type after the java <class> command
- For example you might want to be able to run the program as follows:
  
  java UGLab 200

- meaning run it for 200 timeticks
Using a Command Line Argument

class Main {
    public static void main(String[] args) {
        int nTicks = Integer.parseInt(args[0]);
        UGLab lab = new UGLab();
        Student student1 = new Student("Rudi", 3);
        Student student2 = new Student("Sue", 7);
        lab.add(student1);
        lab.add(student2);
        lab.run(nTicks);
    }
}