In-class java::lang Exercise

Using this repository, you are going to collaborate as a team to complete a translation of a Java program to C++

Moreover, You will be translating "by-hand" some source Java to target C++.

The first team to complete all tasks wins a prize and of course gets glory.

Steps

1. Split up into your project teams.
2. The speaker of each team will fork the repo at https://github.com/nyu-oop/java-lang-in-class
3. Let me know when the fork is prepared and I will add your team members as collaborators.
4. You probably want to strategize a bit, I suggest splitting up into two sub-teams that will pair program.
5. Each sub-team clones the speaker's fork of the xtc-in-class repository.
6. Each sub-team creates a branch on which to work.
7. Open the project in IntelliJ.
   - Open IntelliJ
   - Click 'Open'
   - Browser to the location you cloned to
   - Select the repository directory
   - Check auto-import and both 'download' boxes
   - Make sure Java 1.7 is selected
   - Allow IntelliJ to index the project.
8. Open ./src/test/java/inputs/javalang/Input.java. This is the Java program you will translate to our target language, by hand.
9. Open sbt for this project. Run the sbt command `test:run-main inputs.javalang.Input` and observe the output. Your objective will be to implement this code in C++ such that running the `cpp` command from sbt produces the same results.
10. Look in the output directory. This is where you will put your C++ translation.
    - output.h will contain the declarations of class A and B
    - output.cpp will contain the implementations of class A and B
    - main.cpp will contain the translation of the main method (Tip 1)
11. Implement the data layout and the vtables for each class A and B in output.h (Tip 2)
12. Implement the implementations for each class A and B in output.cpp
13. Implement the main method in main.cpp. (Tip 3)
14. Run both the commands listed in step 9 and screenshot the results.
15. Add the screenshot to your repo.
16. Merge and push to your fork.

Tips

1. Reference the java::lang implementations, you will follow a very similar pattern in your implementations.
2. Note that for the purposes of our translator, we will always have a main.cpp that implements the main method of the Java program. Moreover, we can assume that the class that contains the main method, *only* contains the main method.
3. Remember to convert the Java package to a namespace!
4. Reference https://github.com/nyu-oop/java-lang-1/blob/master/main.cpp for tips on how to invoke your code properly.