

Here are the main links i used.

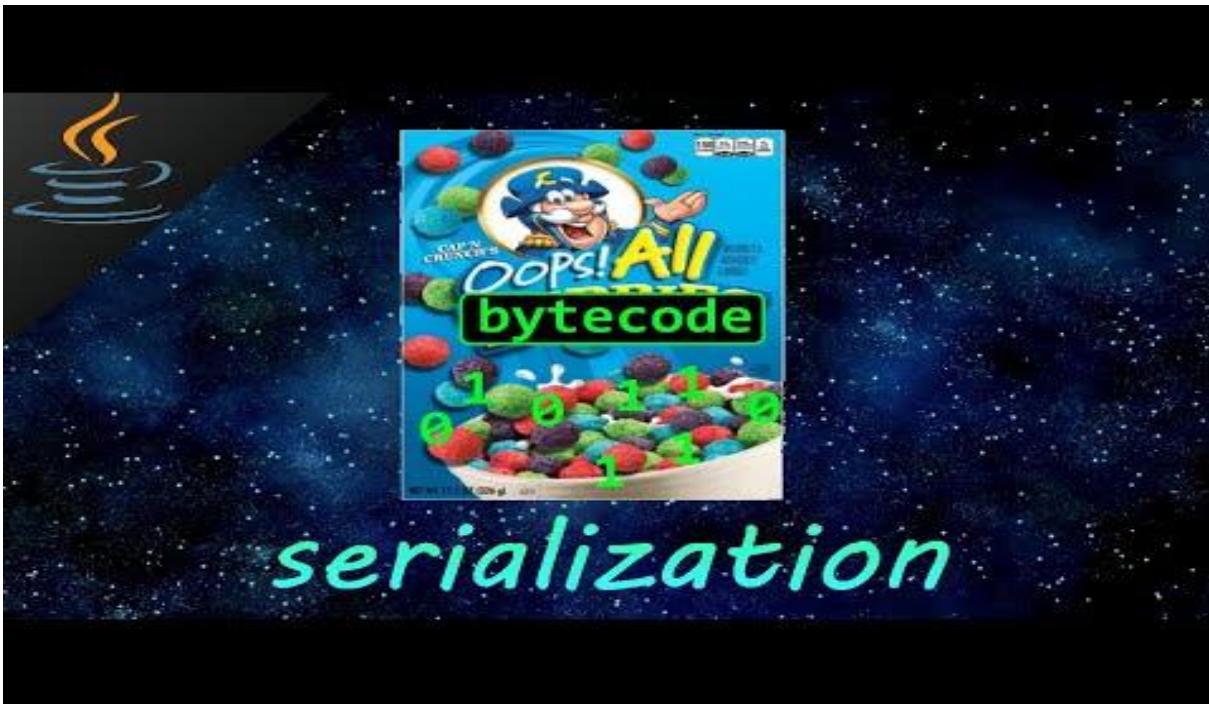
I chose to make MyShapes an abstract class because I never need to create a basic shape by itself. My program only uses real shapes like rectangles and ovals, and an abstract class lets me share common fields (x, y, width, height, color) while still forcing each shape to write its own draw(Graphics g) method. This prevents mistakes, because the compiler makes sure every subclass provides its own drawing code.

Using an abstract class also made the rest of my program simpler. I can store all shapes in one list and call shape.draw(g) without caring which type of shape it is, which is basic polymorphism. Since MyShapes implements Serializable, all subclasses automatically become serializable too, which matches the requirement to save and load the entire list with ObjectOutputStream and ObjectInputStream.

[Abstract Classes and Methods in Java Explained in 7 Minutes - YouTube](#)



[Java serialization is easy! ↗ - YouTube](#)



<https://docs.oracle.com/javase/tutorial/uiswing/components/filechooser.html>

Java custom exceptions



[Checked vs. Unchecked Exceptions in Java Tutorial - What's The Difference?](#)



[Java KeyListener](#) ↗



Notes

- `ObjectOutputStream` takes a Java object (like my list of shapes) and writes it into a binary file. `ObjectInputStream` reads that file and rebuilds the original objects so the shapes can appear again. This lets me save the entire drawing in one step and load it back the same way. It's the main tool Java gives for serialization.
- I check `APPROVE_OPTION` to make sure the user didn't click Cancel in the file chooser. If the user cancels, there is no file to save or load, and trying to continue would cause errors. This check prevents crashes and makes the program behave safely. The rubric also requires checking the result of the dialog.

Ai prompts:

Examples to use the JFileChooser class from the Java Swing library

explain what an abstract class is in Java and when I should use one