Serialization for the new micro service landscape

@leomrlima

CODEMASH 2020

@codemash

WT4 is Serialization?

- What (do you mean by serialization)?
- Why (does it matter now)?
- When (do I have to think about it)?
- Who (is responsible for this)?

If you have to choose…

Restrictions et al

- You want to (or must) use a given framework
- You'll interact with other systems
- You need to store data in a given format
- Your boss to you to

There are only 4 types (really)

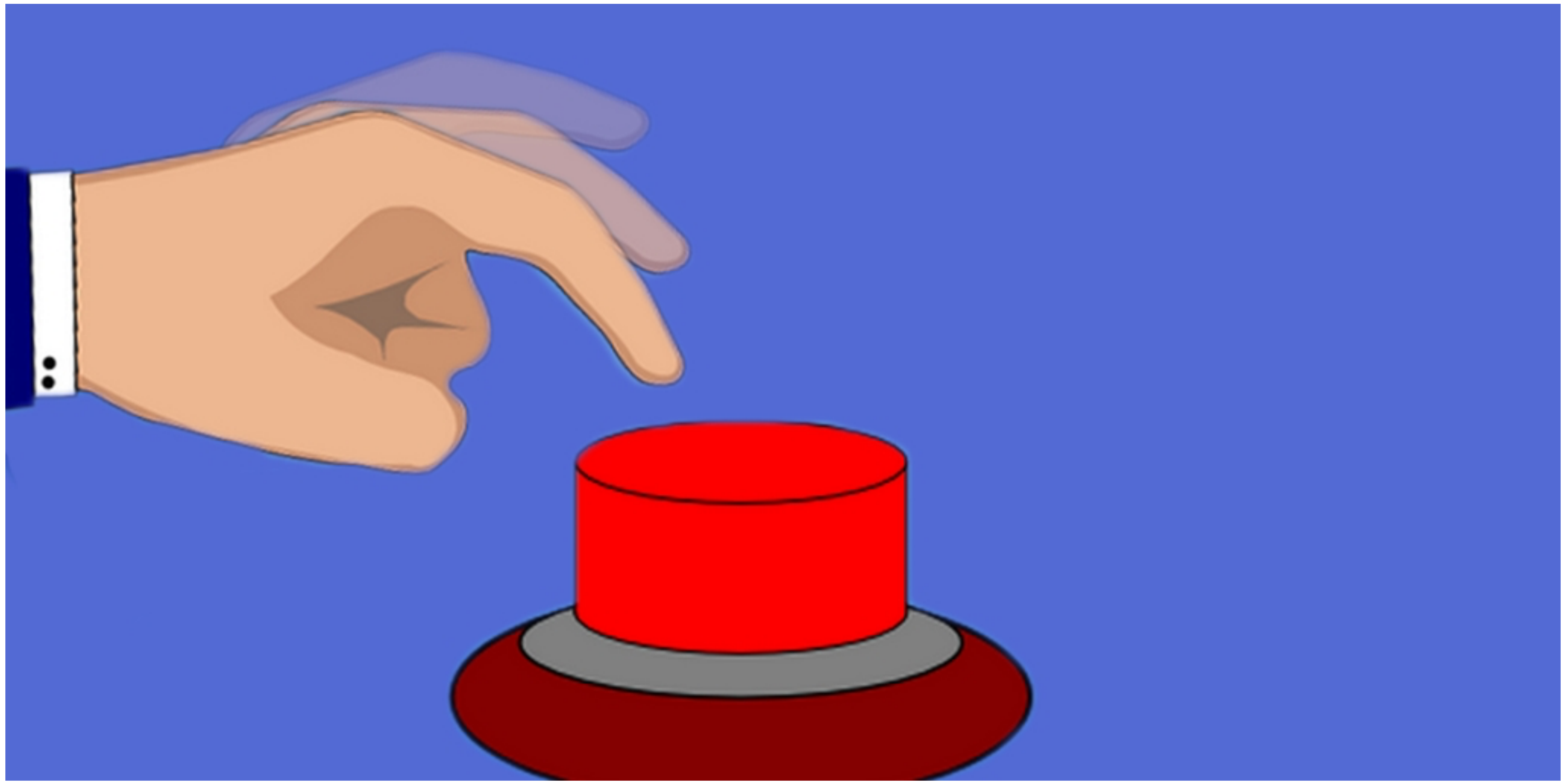
Text x Binary

Does readability by humans matter? When?

Schema-full x Schema-less

When/why to embed the schema in the document?

Ready to choose?...

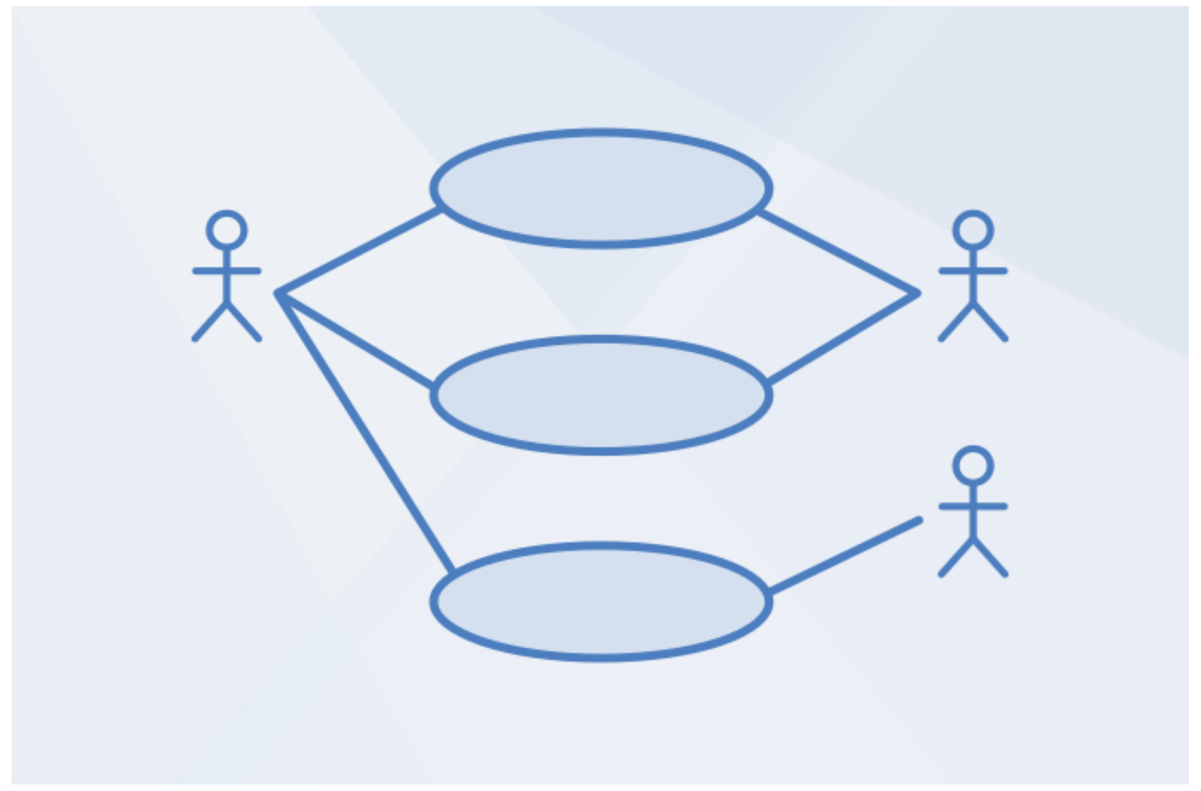


Consequences?

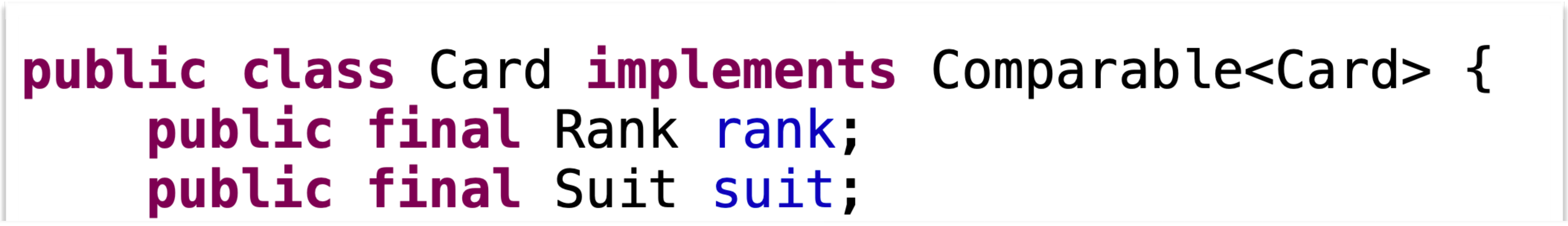
- SIZE of the final byte array;
- SPEED of serialization/deserialization;
- native SUPPORT by non-developer tools;
- COMPATIBILITY with other systems and languages; and
- EASE of development.

Our use case

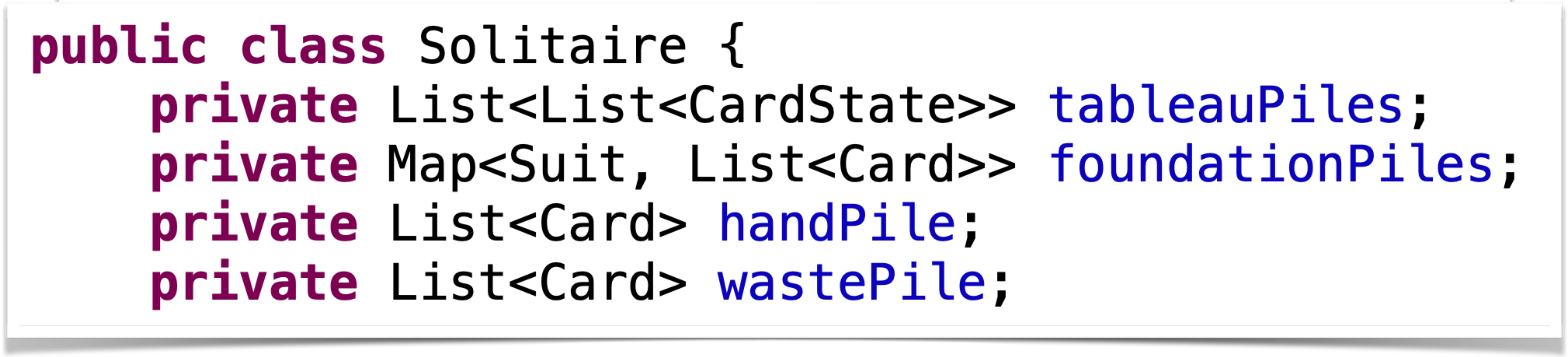
- We're constructing a Card game server
- Our first game is Solitaire
- We have multiple services wanting to exchange data about game state, actions et al



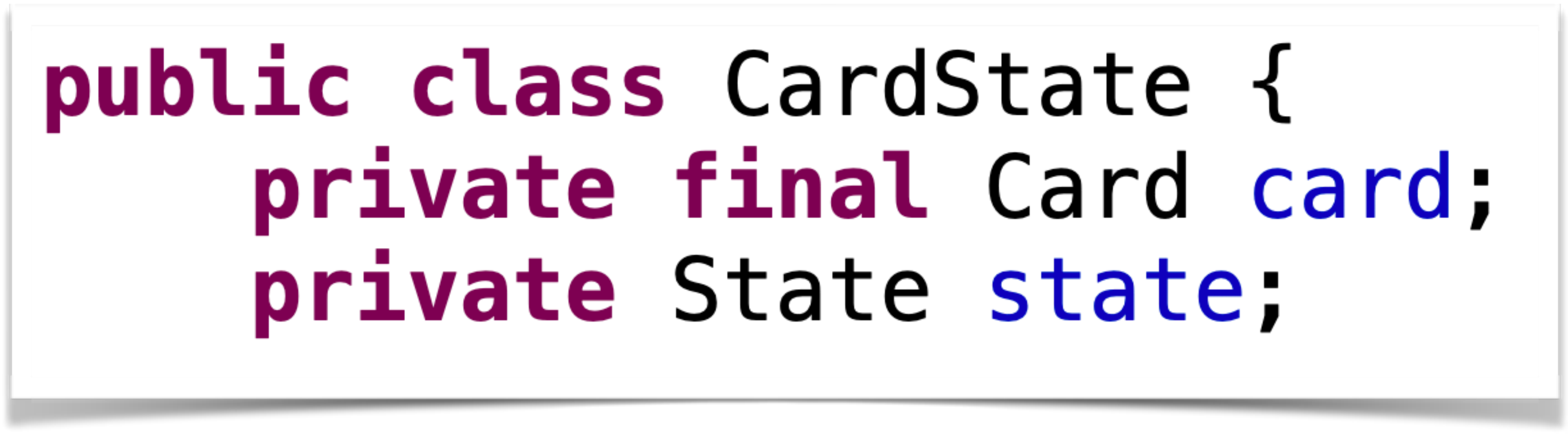
Our use case



```
public class Card implements Comparable<Card> {
    public final Rank rank;
    public final Suit suit;
```

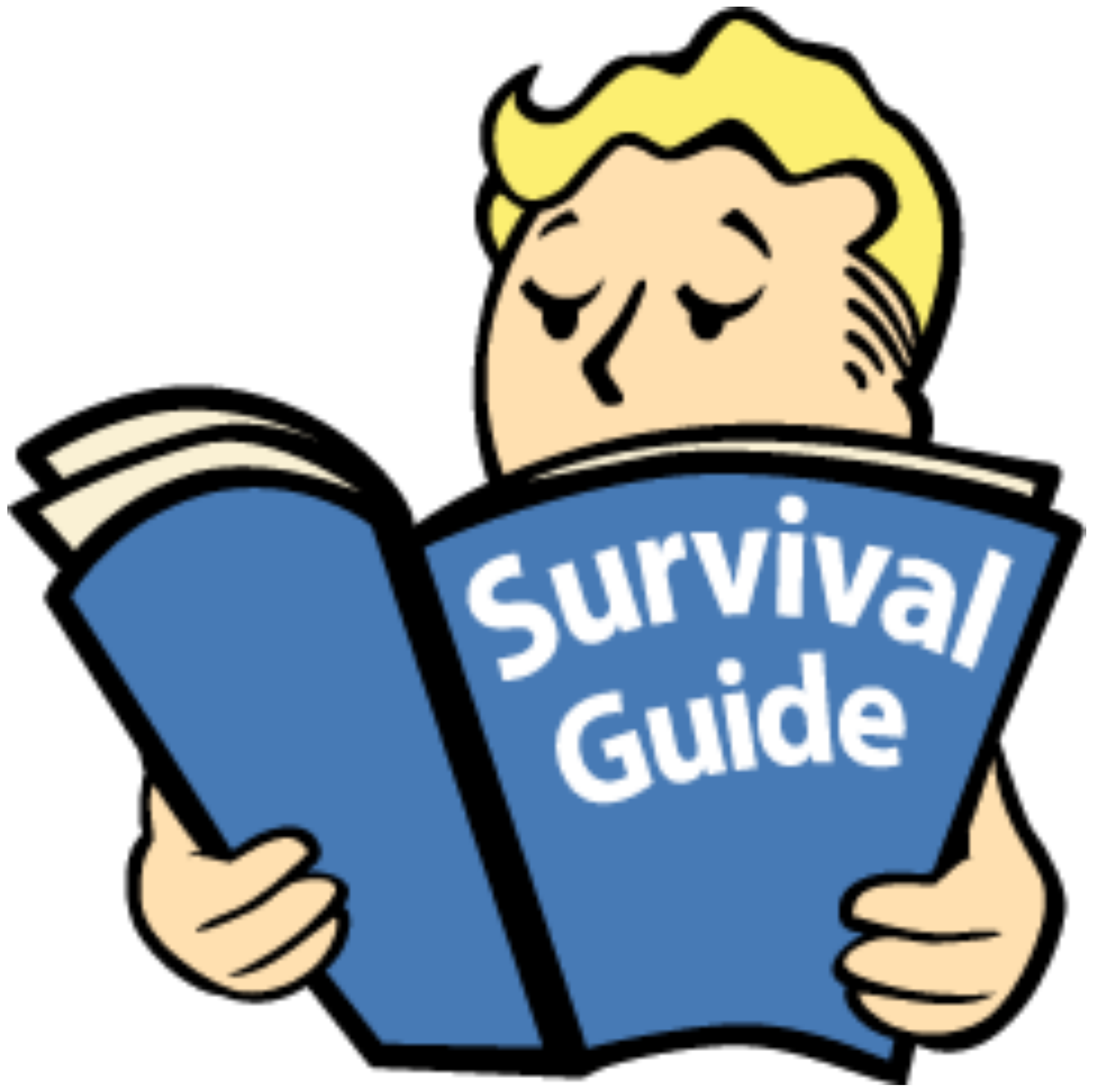


```
public class Solitaire {
    private List<List<CardState>> tableauPiles;
    private Map<Suit, List<Card>> foundationPiles;
    private List<Card> handPile;
    private List<Card> wastePile;
```



```
public class CardState {
    private final Card card;
    private State state;
```

Let’s review our options



Mainstream options

Text & Schema-full

- XML
- JSON

Text & Schema-less

- CSV
- YAML

Binary & Schema-full

- BSON

Binary & Schema-less

- Protobuf
- Avro

How do they compare?

How to compare?

How to compare?

SIZE of the final byte array & SPEED of serialization/ deserialization (and EASE of development!)

Implement a test case and use tools like JMH - Java Microbenchmark Harness

native SUPPORT by non-developer tools & COMPATIBILITY with other systems and languages

The more mainstream you go, the more support you'll have.

CSV

- https://tools.ietf.org/html/rfc4180
- CSV Editor: About 53,800,000 results (0.58 seconds)
- Are you really considering this?!?!

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<solitaire>
   <tableauPiles>
      <pile>
         <list>
            <item state="UP">
               <card rank="KING" suit="DIAMONDS"/>
            </item>
         </list>
      </pile>
      <pile>
         <list>
            <item state="DOWN">
               <card rank="ACE" suit="CLUBS"/>
            </item>
            <item state="UP">
               <card rank="JACK" suit="SPADES"/>
            </item>
         </list>
```

XML

- https://www.w3.org/XML/
- Size of initial game state (unformatted): 3000 bytes
- Size of initial game state (formatted): 4841 bytes
- XML Editor: About 159,000,000 results (0.61 seconds)

<xml />

JSON

```
{
  "tableauPiles": [
    {
      "cards": [
        {
          "card": {
            "rank": "EIGHT",
            "suit": "DIAMONDS"
          },
          "state": "UP"
        }
      ]
    },
    {
      "cards": [
        {
          "card": {
            "rank": "THREE",
            "suit": "DIAMONDS"
          },
```

JSON

- https://www.json.org/json-en.html
- Size of initial game state (unformatted): 3170 bytes
- Size of initial game state (formatted): 7343 bytes
- JSON Editor: About 46,100,000 results (0.50 seconds)

YAML

```yaml
tableauPiles:
- cards:
  - card:
      rank: "JACK"
      suit: "DIAMONDS"
    state: "UP"
- cards:
  - card:
      rank: "THREE"
      suit: "CLUBS"
    state: "DOWN"
  - card:
      rank: "TEN"
      suit: "CLUBS"
    state: "UP"
- cards:
  - card:
      rank: "FOUR"
      suit: "DIAMONDS"
```

YAML

- https://yaml.org
- Size of initial game state: 3768 bytes
- YAML Editor: About 2,490,000 results (0.53 seconds)

BSON

- http://bsonspec.org
- Size of initial game state: 3839 bytes
- BSON Editor: About 116,000 results (0.46 seconds)

Protobuf

- https://developers.google.com/protocol-buffers
- Size of initial game state: 489 bytes
- Protobuf Editor: About 187,000 results (0.53 seconds)
- Major drawback: EASE of use

protobuf
Protocol Buffers

Avro

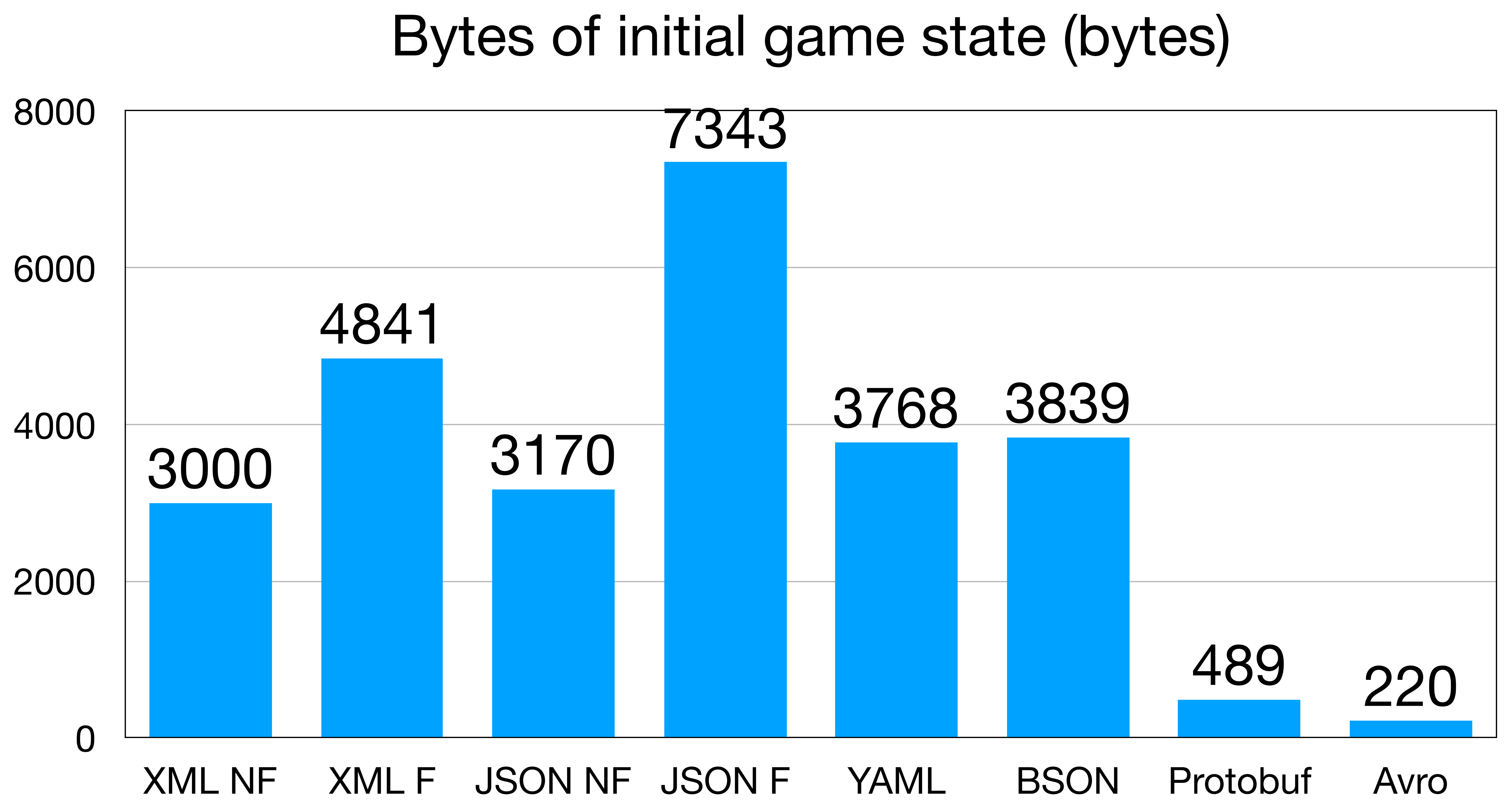
- https://avro.apache.org
- Size of initial game state: 220 bytes
- Avro Editor: About 818,000 results (0.53 seconds)
- Major drawback: EASE of use

By Source (WP:NFCC#4), Fair use, https://en.wikipedia.org/w/index.php?curid=61196747

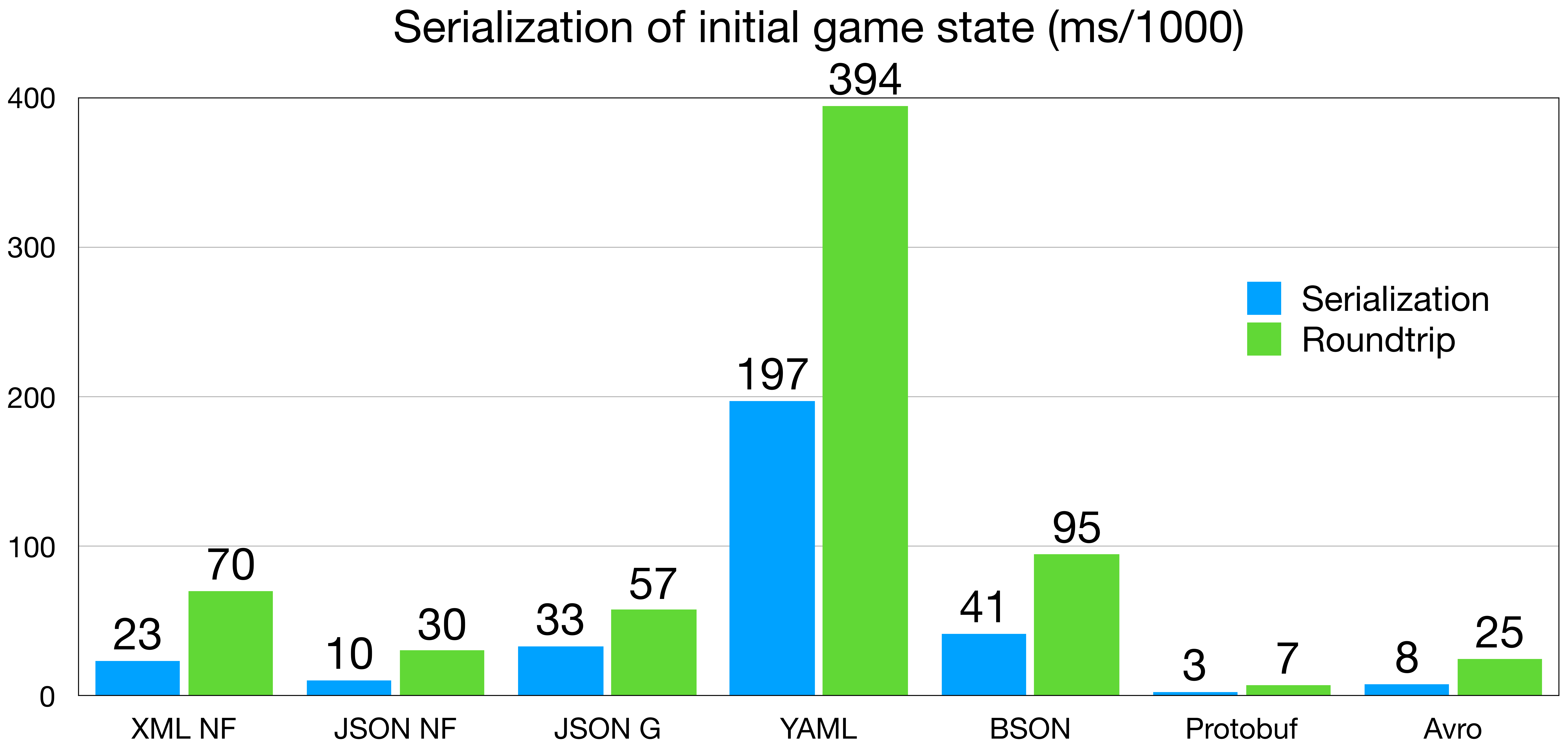
The tests

- Implemented in Java to compare size & speed
- Source code at https://github.com/leomrlima/serialization
- Used libraries and plugins compatible with the Jackson project when possible; also used Google Gson
- Implementing the use case highlights the good/bad of each approach regarding the non-numeric aspects!

How do they compare?



How do they compare?



KEEP
CALM
AND
LET'S
RECAP
KeepCalmAndPosters.com

In short…

- **XML** is verbose but it’s well supported
- **JSON** seems to have the right balance
- **YAML** can get complicated for complex structures
- **BSON** doesn’t seem to improve over text (in size)
- **Avro and Protobuf** improves size & speed in detriment of ease of use

Yes, BUT...

There’s no golden rule

Unfortunatelly, you HAVE to try it yourself

Hope this example helps to guide your way through it!

Thanks!

- @leomrlima
- linkedin.com/in/leomrlima
- leomrlima@gmail.com
- https://github.com/leomrlima/serialization