Saving Object State

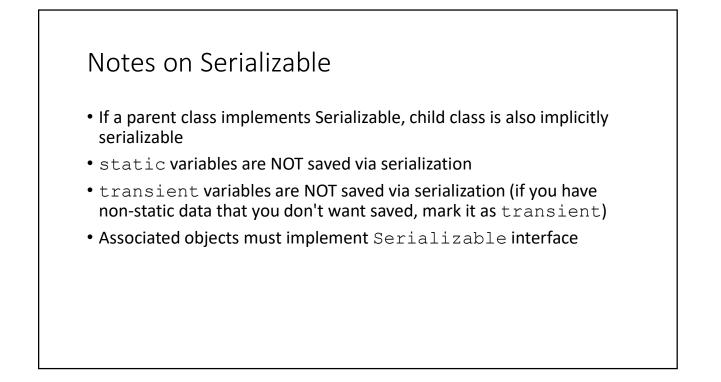
CSCI 2300

In your team project you will need to

- Save some state (high scores, unfinished game state, etc)
- Load the state when application is started

Serialization A process of converting an object to a byte stream The byte stream can be written to ObjectOutputStream for Writing byte stream to a file Sending byte stream over a network java.io.Serializable interface Marker interface – has no methods or variables Used to mark java classes to as capable of being saved

Example	<pre>import java.io.Serializable; public class Player implements Serializable { private int numWins; private String name; public Player(String n, int w) </pre>
1	<pre>{ numWins = w; name = n; } } filename) w FileOutputStream(filename); ew ObjectOutputStream(file);</pre>



```
import java.io.Serializable;
public class GameState implements Serializable
{
   private Player p1;
```

```
public class Player
  private String name;
  private int numWins;
```

{

}

public class GameBoard implements Serializable

private [][]GamePiece;

private GameBoard board;

private Player p2;

{

GameState objects will not get serialized properly. Why?

A. Because instance variables of GameState class are private

- B. Because Player class is not Serializable
- C. Because GameBoard has a 2-dimentional array
- D. Because GameState does not implement the methods of Serializable interface

```
import java.io.Serializable;
public class GameState implements Serializable
{
    private Player p1;
    private Player p2;
    private transient GameBoard board;
}
```

```
public class Player
implements Serializable
{
```

private String name;
private int numWins;

}

```
public class GameBoard implements Serializable
{
```

private [][]GamePiece;

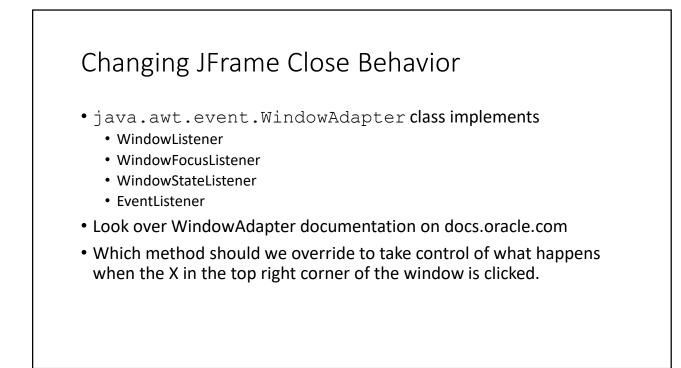
Which variables of GameState will get saved during serialization?

A. p1, p2, and board B. p1 and p2 C. board D. None

Deserialization

- A process of creating an object from a byte stream
- Constructor of an object being deserialized is never called

```
public class GameDriver
{
    private Player p1;
    private Player p2;
    public void loadFromFile(String filename) throws Exception
    {
        FileInputStream file = new FileInputStream(filename);
        ObjectInputStream in = new ObjectInputStream(file);
        in.readObject(p1);
        in.readObject(p2);
        in.close();
        file.close();
    }
}
```



```
Example
public class GameWindowAdapter extends WindowAdapter
{
    // some data
    @Override
    public void windowClosing(WindowEvent e)
    {
        // do what needs to be done
        System.exit(0); // terminate the application
    }
}
```