Java Enums

Why have Choice.Yes and Choice.No?
Old Way

- Enumerated types consist of a set of named values
- Example: Font has BOLD, ITALIC rather than 1, 2

```java
// From class Font:
public static int BOLD = 1;
public static int ITALIC = 2;
```

- It's easier to remember Font.BOLD than 1
- Font.BOLD in action

```java
JButton button = new JButton("Courier 24 point Bold");
button.setFont(new Font("Courier", 24, Font.BOLD));
```
An int is an int

- private static ints are also know as "glorified integers"
- This can lead to problems
  - Can you read the text in the button with a point size of 1?

Since Font.BOLD is just an int you can pass in any other int value, like 2314

And since the style Font.BOLD is compatible with point size (both int type parameters), you get one (1) point font like above

Swap the arguments to get what was intended see next slide
Would be nice to have different type

- Since `Font.BOLD` is just an int, you can pass in any other int value, like 2314 or -123
- And since the style `Font.BOLD` is compatible with point size (both int type parameters), you get one (1) point font like above
- Swap the arguments to get what was intended

```java
button.setFont(new Font("Courier", 24, Font.BOLD));
button.setFont(new Font("Courier", Font.BOLD, 24));
```
New Way

- Use the Java enum type like this

```java
public enum FontStyle { BOLD, ITALIC };
```

- The previous attempt to construct a Font would preferable be a compile-time error

- With an enum type, we get an improved constructor

```java
public Font(String name, FontStyle style, int size)
```
An Enum used in GameTree

- **playerSelected** using a String parameter could result in code like this

```java
public void playerSelected(String yesOrNo) {
    if (yesOrNo.equals("Yes"))
        ; // go left
    else if (yesOrNo.equals("No"))
        ; // go right
    else
        JOptionPane.showMessageDialog(null,
                                        "Contact vendor!");
}
```

- Just hope the user never types "Y", "yes" or "y"
  - and the programmer used **equals**, not **==**
**Must use a Choice argument**

- Here is the actual design of `playerSelected`

```java
/**
 * Ask the game to update the current node by going left for Choice.yes or right for Choice.no Example code:
 * theGame.playerSelected(Choice.Yes);
 *
 * @param yesOrNo
 */

public void playerSelected(Choice yesOrNo) {
    if (yesOrNo == Choice.Yes)
        // go left
    if (yesOrNo == Choice.No)
        // go right
}
```