Generics & Exceptions

Fixing our Stacks and Queues

Current Status

Stack of integers

Queue of characters

To expand this for integer, character, doubles, Strings, Coordinates, Cars, Animals, JFrame, JPanel, etc…

9+ classes of Stack; 9+ classes of Queue

Ugh

Options?

Could use inheritance…

Stack of Objects

but: no safety! *(did I only push Car objects?)*

Java loves safe programming!

but: would always need to "cast" the popped value

Options: Generics

Create Stack and Queue - but leave object type undefined

Will look like this:

```
public class Stack<Element> { …
```

Will be instantiated like this:

```
Stack<Coordinate> myStack
    = new Stack<Coordinate>(25);
```

Generics

"allow a type or method to operate on objects of various types while providing compile-time type safety"

So what?

I can write ONE Stack for all objects and the compiler will still be able to check for errors

```
Stack badIdea = new Stack( );
badIdea.push ("Hello");
badIdea.push ("World");
String t1 = (String) badIdea.pop( );
String t2 = (String) badIdea.pop( );
```

```
Stack<String> st = new Stack<String>( );
st.push ("Hello");     //type checked!
st.push ("Hello");
String s1 = st.pop( ); //No casting!
String s2 = st.pop( );
```

Maze Runner

Need a stack of (future) choices

Coordinates

Generic type

Allow a class to manipulate a variety of types without having to rewrite/duplicate the class

Java Collections do this

ArrayList, etc

Java Generics

When defining the class, we “tag” it with a generic type

- `public class Stack<Element>`

“Element” is my choice and could be most anything - Sometimes simply ‘E’

Throughout the code…

Instead of specific type, use generic Element

```
package myADT;

/**
 * Stack class implements basic push, pop, and isEmpty
 * NOTE: There is currently no validation when pushing onto a full stack
 * NOTE: There is currently no validation when popping an empty stack
 * @author Stephen Hall
 * @since  Jan 24,2015
 */
public class Stack<Element> {
    private Element[ ] arrStack;    //stack as an array
    private int top;                //current top …
    private final int MAX_SIZE;

    /**
     * Creates a stack of 10
     */
    public Stack ()
    {
        this (10);
    }

    /**
     * Creates a stack of the given maximum size
     * @param max
     */
    @SuppressWarnings("unchecked")
    public Stack (int max)
    {
        if (max > 0)
            MAX_SIZE = max;
        else
            MAX_SIZE = 10;

        arrStack = (Element[ ]) new Object[MAX_SIZE];
        top = -1;
    }
```

```
    /**
     * Returns true if the stack has any valid items on it
     * @return
     */
    public boolean isEmpty()
    {
        return top < 0;
    }

    /**
     * Adds the given integer to the top of the stack
     * @param value
     */
    public void push (Element value)
    {
        if (top < MAX_SIZE)
            return;
        top++;
        arrStack[top] = value;
    }

    /**
     * Removes the top item on the stack
     * @return
     */
    public Element pop ()
    {
        if (isEmpty())
            return null;
        Element val = arrStack[top];
        top--;
        return val;
    }
```

Beware

Generics work with Objects only

not fundamental types (int, double, char, etc)

must use “wrapper” classes for those

Integer, Double, Character, etc