CECS 277 – Lecture 1 – Classes Review

Classes – A class is a blueprint for an object. It specifies what an object has access to and what it does. Once it's defined, one or more objects can then be made from the class.

A class is made up of data members and methods. Data members store information about the object, and methods are used to access or modify the data members and to add functionality to the object.

Classes are particularly useful in the maintenance of larger projects. They cut down on excess code by being extendable and reusable. They encapsulate code by keeping related code together and the data protected. They ensure the programmer that once the class has been written and tested, that it doesn't need to be retested each time code is changed somewhere else. Plus, once a class is written and tested, it can be used in other programs.

Example – Dice Class – A Die has several features and functions.

Features: Stores the number of sides on the die, and the value rolled.

Functions: The die may be rolled, set to a particular value, or viewed to see the value of the die.

Once the Die class is created and tested, it can be used in any number of programs that use one or more dice without worrying about changing values or modifying code for a particular game's functionality. If new functions are required for a Die, it can be added to the Die class directly (if the functionality is needed for all future dice), or the Die class may be extended to create new and different types of Dice while still preserving the original Die class.

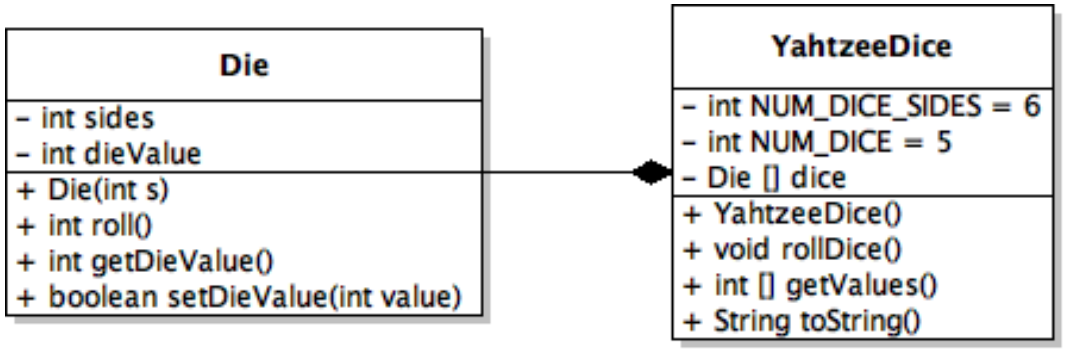
The YahtzeeDice Class - Here we will use the Die class to create an array of five dice that will be used in a Yahtzee type game.

Features: Defines the number of dice to roll, the number of sides of each die. Also creates storage for each of the five dice objects.

Functions: All of the dice may be rolled, and their values viewed.

Once the YahtzeeDice class has been created and tested, it too can be used in any games that require 5 six-sided dice, or be extended to create new types of YahtzeeDice classes.

A UML Diagram of the relationship between the Die and YahtzeeDice classes –



```java
/**
 * Die Class - representation of a single die
 * @author Shannon Cleary
 */
public class Die {
	/** The number of sides on this die */
	private int sides;
	/** The value of this die */
	private int dieValue;
	/**
	 * Constructor - rolls to give the die an initial value
	 * @param s sets the number of sides of this die
	 */
	public Die( int s ) {
		sides = s;
		roll();
	}
	/**
	 * Rolls the die
	 * @return the die's value
	 */
	public int roll() {
		dieValue = (int)( Math.random() * sides ) + 1;
		return dieValue;
	}
	/**
	 * Retrieve the die's value
	 * @return the die's value
	 */
	public int getDieValue() {
		return dieValue;
	}
	/**
	 * Allows the value of the die to be set to a particular value
	 * as long as it is within range of the number of sides
	 *
	 * @param value  the value the die will be set to
	 * @return true if the value could be set, false otherwise
	 */
	public boolean setDieValue( int value ) {
		if( value > 0 && value <= sides ) {
			dieValue = value;
			return true;
		} else {
			return false;
		}
	}
}
```

```java
/**
 * YahtzeeDice Class - Set of 5 dice that may be rolled
 * @author Shannon Cleary
 */
public class YahtzeeDice {
	/** The number of sides for each die */
	private final int NUM_DICE_SIDES = 6;
	/** The number of dice used in this game */
	private final int NUM_DICE = 5;
	/** The set of dice for the game */
	private Die [] dice;
	/** Constructor - initializes each of the dice */
	public YahtzeeDice() {
		dice = new Die[NUM_DICE];
		for( int d = 0; d < dice.length; d++ ) {
			dice[d] = new Die( NUM_DICE_SIDES );
		}
	}
	/** Rolls all of the dice */
	public void rollDice() {
		for( int d = 0; d < dice.length; d++ ) {
			dice[d].roll();
		}
	}
	/**
	 * Retrieve the values of the dice
	 * @return the values of the dice as an array of integers
	 */
	public int[] getValues() {
		int [] values = new int[NUM_DICE];
		for( int d = 0; d < dice.length; d++ ) {
			values[d] = dice[d].getDieValue();
		}
		return values;
	}
	/**
	 * String representation of the Yahtzee object.
	 * @return a string representation of the dice
	 */
	@Override
	public String toString() {
		String values = "";
		for( int d = 0; d < dice.length; d++ ) {
			values += "D"+(d+1)+": "+dice[d].getDieValue()+" ";
		}
		return values;
	}
}
```

```java
/**
 * YahtzeeGame - game that rolls 5 dice two times to test to make
 * sure the YahtzeeDice class works properly.
 * @author Shannon Cleary
 */
public class YahtzeeGame {
	public static void main(String[] args) {

		//create our 5 dice
		YahtzeeDice dice = new YahtzeeDice();

		//instantiate the dice and print their values
		System.out.print("First Roll: ");
		for( int i : dice.getValues() ) {
			System.out.print( i + " " );
		}
		System.out.println();

		//roll the dice and print values using toString()
		dice.rollDice();
		System.out.print( "Second Roll: " );
		System.out.println(dice);
	}
}

/* Output:
 *
 * First Roll: 1 1 2 6 2
 * Second Roll: D1: 4 D2: 1 D3: 4 D4: 5 D5: 5
 */
```