Grant Elgin
9/4/13
HW 1 Java term definitions.

A: protected
protected is an access level modifier to be used with methods or variables in a class. If a method is declared protected, that method can only be accessed by members of the same package (the same behavior as not declaring a modifier) but it can also be accessed by a sub-class of its class in another package.

B: static
The keyword static is used to declare variables that are shared for all instances of the class. Static variables reference a single object in a specific memory location. All instances of the class use the same original copy of the static variable. If variables are not declared static, each instance of that class will create its own copy of those non-static variables that will have their own location in memory.

C: new
The keyword new is used to instantiate a copy of a class. new is the keyword that allocates the required memory for a class.

D: interface
An interface provides method declarations and constant declarations that are required to be implemented in classes that extend the interface. Only the declaration of the methods are in an interface. Method bodies are not defined. System architects use interfaces to outline required methods for other team members to implement.

E: abstract
Classes declared as abstract can include method declarations without method bodies (similar to an interface but declared abstract: abstract boolean isASquare(Shape obj);) as well as implemented methods (has method body).

```java
/**
 * Grant Elgin
 * Sep 5, 2013
 * CS 342 HW1
 */

import java.util.Random;
import java.util.Scanner;

public class ImThinkingOfANumber {

    private static int randomNumber = 0;
    public static final int MIN_NUMBER = 1;
    public static final int MAX_NUMBER = 1000;
    private static int numberOfTries = 1;

    public static void main(String[] args) {
        // display name of program and exit method
        // generate a random number
        initializeGame();

        // verify a random number has been generated
        // if we have a random number, display instructions and let the user
            guess
        if (getRandomNumber() != 0) {
            System.out.println("I'm thinking of a number between 1 and 1000.
                Guess by entering a number and pressing return.");
            letUserGuess(getRandomNumber());
        }
        else {
            System.out.println("Uh oh! Something went wrong!");
            System.exit(0);
        }
    }

    private static void initializeGame() {
        System.out.println("ImThinkingOfANumber v1.0\nType '9999' to exit.");
        generateRandomNumber();
    }

    private static int generateRandomNumber() {
        //using java.util.Random class to generate random number
        Random rand = new Random();

        // nextInt excludes MAX_NUMBER so add 1 to include it as an option
        int randomNum = rand.nextInt((MAX_NUMBER - MIN_NUMBER) + 1);

        // after the random number is set here, it will be verified in main
            method before letUserGuess is called
        setRandomNumber(randomNum);

        return randomNum;
    }
```

```
    private static void letUserGuess(int r) {
        Scanner sc = new Scanner(System.in);

        // if user input is not an int, catch the exception and allow the user to
            try again
        // if it is an int, continue to compareGuess.
        try {
            int guess = sc.nextInt();
            compareGuess(guess);
        }

        catch (Exception e) {
            System.out.println("Please guess a number between " + MIN_NUMBER +
                "and " + MAX_NUMBER);
            letUserGuess(r);
        }

    }

    private static void compareGuess(int guess) {

        // each guess will increment numberOfTries by 1
        // check to see if guess is lower, higher, the exit string 9999, or a
            match
        if (guess < getRandomNumber()) {
            System.out.println("Too low. The number I'm thinking of is higher.
                Guess again.");
            setNumberOfTries(getNumberOfTries() +1);
            letUserGuess(getRandomNumber());

        }
        else if (guess > getRandomNumber()) {
            if (guess == 9999) {
                System.out.println("Thanks for playing!");
                System.exit(0);
            }
            else {
            System.out.println("Nice try, but the number I'm thinking of is lower
                than that. Guess again.");
            setNumberOfTries(getNumberOfTries() + 1);
            letUserGuess(getRandomNumber());
            }

        }
        else if (guess == getRandomNumber()) {
            System.out.println("That's right! You guessed it!");
            System.out.println("It took you " + getNumberOfTries() + " tries to
                guess the number. ");
            System.exit(0);
        }

    }
```

```
    public static int getRandomNumber() {
        return randomNumber;
    }

    public static void setRandomNumber(int randomNumber) {
        ImThinkingOfANumber.randomNumber = randomNumber;
    }

    public static int getNumberOfTries() {
        return numberOfTries;
    }

    public static void setNumberOfTries(int numberOfTries) {
        ImThinkingOfANumber.numberOfTries = numberOfTries;
    }

}
// end of Class ImThinkingOfANumber
```