COP 3022
Final Exercises

```
public class Default1 {
   private int count;

   public int getCount() {
      return count;
   }

public static void main(String[] args) {
   Default1 d = new Default1();
   System.out.println(d.getCount());
  }
}
```

Compiles?

```
public class Default2 {
   private int count;

   public Default2(int count) {
      this.count = count;
   }

   public int getCount() {
      return count;
   }

public static void main(String[] args) {
   Default2 d = new Default2();
   System.out.println(d.getCount());
}
```

Compiles?

```
public class Multi {
   private int number;

   public Multi(int number) {
      this.number = number;
   }

   public Multi() {
      // set number to 100
   }
}
```

Complete!

```
public class Super {
   private int age;

   public Super(int age) {
      this.age = age;
   }
}

public class Sub extends Super {
   private String name;

   public Sub(int age, String name) {
      // Fill in the correct code
   }
}
```

Complete!

```
public class Ship {
   private String name;

   public Ship(String name) { this.name = name; }

   @Override
   public String toString() {
      return "[Ship] " + name; }
   }

public class CruiseShip extends Ship {
   private int maxPassengers;

   // Constructors, etc omitted

   @Override
   public String toString() {
       // Formatted as
       // [Ship] Titanic max: 1200
   }
}
```

Complete!

```
public class Employee {
   private int salary;

   public Employee() { this.salary = 1000; }

   @Override
   public String toString() {
      return "I got " + salary + "!";
   }
}

public class WeirdEmployee extends Employee {
   @Override
   public String toString()
       { return "GoooGoooGooo!!!";
}

public static void main(String[] args) {
   Employee e = new WeirdEmployee();
   System.out.println(e);
}
```

Result?

```
public class Person {
   public void print() {
      System.out.println("I am a person");
   }
}
public class Nobility extends Person {
   public void print() {
      super.print();
      System.out.println("I am a noble person");
   }
}
public class King extends Nobility {
   public void print() {
      super.print();
      System.out.println("I am a king");
   }
}
public static void main(String[] args) {
   Person p = new King();
   p.print();
}
```

Result?

```
public abstract class AbstractC {
   private String s;

   public AbstractC() {
      s = "hello";
   }
   public abstract void put();
   public abstract void show();
}

public class ConcreteC extends AbstractC {
   @Override
   public void put(String s) {
      this.s = s;
   }
   @Override
   public void show() {
      System.out.println(s);
   }
}
```

Compiles?

```
public interface Doable {
   void doIt();
}

public class Worker implements Doable {
   @Override
   public void doIt() {
      System.out.println("I'm doing it!!!");
   }
}
```

Compiles?

```
public class TopException extends Exception {
   public TopException(String message) {
      super(message);
   }
}

public class LowException extends TopException {
   public LowException(String message) {
      super(message);
   }
}
public static void main(String args[]) {
   try {
      // Code that throws a "LowException"
   } catch (TopException e) {
      System.out.println("Top exception");
   } catch (LowException e) {
      System.out.println("Low exception");
   }
}
```

Compiles?

```
public class Car {
   private String make;
   private String model;
   private int hp;

   public Car(String make, String model, int hp) {
      this.make = make;
      this.model = model;
      this.hp = hp;
   }

// setters/getters omitted

  public String toString()
  {
    return "make = " + make + " model = " + model
         + " hp = " + hp;
  }
}
```

- Write a class **Garage**
- Instance fields: an **ArrayList** of **Car**s, others at your discretion
- Default constructor - initializes instance variable(s)
- A method to add a **Car** (instance) to the **Garage**
- **toString()** creates and returns a String that displays all the **Car**s in the **Garage**
- An additional constructor **public Garage(String fileName)**. that reads one car per line from a text file, each line being **make, model, hp**.Example:

```
VW Beetle 35
Ford Focus 90
Toyota Corrola 75
```

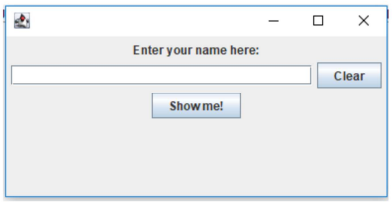
```
public class WrappersWithArray {
   private static final int MAX = 10;
   private int[] numbers;
   private int occupied;

   public WrappersWithArray() {
      numbers = new int[MAX];
      occupied = 0;
   }

   public void add(int x) {
      numbers[occupied++] = x;
   }

   public void showAll() {
      for (int i = 0; i < occupied; i++) {
         System.out.print(numbers[i] + " ");
      }
   }
}
```

Rewrite with ArrayList!



Develop a class that extends **JFrame** that looks like this when made visible.**[4]**

Write an **ActionListener** *as an inner class* that

- resets the Text Field when the "Clear" button is clicked **[2]**
- prints "So, you are _____" into System.out, where the contents of the Text Field is displayed (instead of the underlines) **[2]**

Write a recursive method **center** that takes in a **String** and returns the central character(s) of this string. If the string has an odd length, it's the character in the middle, if it's even the two characters in the middle.

Examples:

- center("hazard") is "za"
- center("Strings") is "i"
- The center of the empty String is the empty String
- The method must work for any size String!