GUI — EventHandling
— MouseEvents

Interface
— Before JavaFX

Interface and Polymorphism:

Consider the following Interface definition

```
Public interface AnInterface
{
    Public String howToDo();
}
```

- Defile two classes ClassA and classB which implement AnInterface interface.

- Suppose you are given the following definitions:

```
ClassA orange = new ClassA();
AnInterface chicken;
ClassB sub = new ClassB();
Rectangle box = new Rectangle(5, 10, 20, 30);
```

Which of the following assignments are legal?
For the illegal statements, specify if the error is caught by the compiler or during the execution.

- chicken = orange;
- orange = chicken;
- chicken = sub;
- orange = (ClassA)chicken;
- orange = box;
- orange = (ClassA)box;

```
public abstract class Person {
   private String name;
   private String address;

   public Person(String aName, String anAddress) {
      name = aName;
      address = anAddress;
   }
   public abstract void printInfo();
   @Override
   public String toString(){
      return getClass().getName() + "[" + name +
          "," + address + "]";
   }
}
```

```
public class Employee extends Person {
   private double salary;
   public Employee(String name,
          String address, double salary)
   {
      super(name, address);
      this.salary = salary;
   }
   @Override
   public void printInfo() {
      System.out.println("Employee with " +
          salary + " salary");
   }
}
```

```
public class Student extends Person {
   private int totalCredits;
   public Student(String name, String address)
   {
      super(name, address);
      totalCredits = 0;
   }
   @Override
   public void printInfo(){
      System.out.println("Domestic Student");
   }
   @Override
   public String toString(){
      return super.toString() + "[credits=" +
                              totalCredits + "]";
   }
}
```

```
public class InternationalStudent
                        extends Student{
   public InternationalStudent(
          String name, String address)
   {
      super(name, address);
   }
   @Override
   public void printInfo(){
      System.out.println
          ("International Student");
   }
}
```

```
import java.util.ArrayList;
public class College {
   private ArrayList<Person> people;
   public College(){
      people = new ArrayList<Person>();
   }
   public void addPerson(Person aPerson){
      people.add(aPerson);
   }
   public void printInfo() {
      for(Person p: people) {
         p.printInfo();
      }
   }
   @Override
   public String toString(){
      String result = getClass().getName()+ "\n";
      for(Person p: people) {
         result += "\t" + p + "\n";
      }
      return result;
   }
}
```

```
public class Tester {
   public static void main(String[] args){
      College langara = new College();
      langara.addPerson(new Student("Alice", "100 49th Ave."));
      langara.addPerson(new Employee("June", "200 Granville", 60000));
      langara.addPerson(new InternationalStudent("Jag",
                              "120 Ontario st"));
      System.out.println(langara);
      langara.printInfo();
   }
}
```

1. Draw a UML class diagram showing the relationships between the Person, Employee, Student, InternationalStudent and College classes.

2. Specify if the following statements are correct or not.

```
Person p1 = new Person("Mike" , "Main st");

Person p2 = new Employee("Jo", "Granvill" , 70000);

Person p3 = new InternationalStudent("Jag" , "Oak st");

Student s1 = new InternationalStudent("Jashan" , "West 48th");

InternationalStuden s2 = new Student("Jo", "Ontarion st");

Student s3 = new Employee("Billy", "East 30th ave", 50000);
```

3. What is the output of the Tester program? Please show your work.

```
public class ExceptionsTesting {
    public static void main( String args[] ) {
            try {
                func1();
                System.out.println("List of Flowers");
             }
             catch ( Exception exception ) {
                 System.out.println( "Tulips" );
                 try {
                         func2();
                 }
                 catch ( Exception e ) {
                          System.out.println("Snowdrop");
                 }
                 finally{
                         System.out.println("Cherry Blossom");
                 }
             }
             finally {

                 System.out.println("Lily");
             }

              System.out.println("Rhododendron");
        }

        public static void func1() throws Exception {

           try {
                System.out.println( "Daisy" );
                throw new Exception();
           }

           catch ( Exception exception ) {
                System.out.println( "Azalea" );
                throw exception;
           }

           finally {
                System.out.println( "Rose" );
           }
        }
        public static void func2() throws Exception {
           func3();
        }

         public static void func3() throws Exception {
                 throw new Exception("Begonia");
         }
}
```

```
public class TestingExceptions {
    public static void main( String args[] ) {
        try {
            function1();
            System.out.println("List of flowers");
        }

        catch ( Exception exception ) {
            System.out.println( "Lily of the Valley" );
        }

        finally {
            try {
                function2();
                System.out.println("Begonia");
            }

            catch ( Exception e ) {
                System.out.println( e.getMessage());
            }
        }
    }

    public static void function1() throws Exception {
        try {
            System.out.println( "Rhododendron" );
            throw new Exception("Tulip");
        }

        catch ( Exception exception ) {
            System.out.println( "Azalea" );
            throw exception;
        }
    }

    public static void function2() throws Exception {
        throw new Exception( "Foxglove" );
    }

}
```