Reminders

This test is closed-notes and closed-book. However, you are allowed to bring 1 page (8.5 X 11) of notes (both sides). Your notes must be your own, they must be hand written, and they must be turned in with your test. This test is to be done individually, and you are not to exchange or share your notes with other students during the test.

If you need more space, use the back of a page. Note when you do that on the front.

This test is timed. Your test will not be graded if you try to take more than the time allowed. Therefore, before you begin, please take a moment to look over the entire test so that you can budget your time.

For diagrams and programs, clarity is important; if your diagrams or programs are sloppy or hard to read, you will lose points. Correct syntax also makes some difference.

There are 100 points all.

1. (2 points) Software development is difficult, time-consuming, and costly. Which of the followings is less likely to be one of the main causes of the problems in software development?

   (a) Complexity
   (b) Longevity and evolution
   (c) High user expectation
   (d) New programming languages

2. (2 points) There are many desirable qualities of software systems, including usefulness, timeliness, reliability, maintainability, reusability, user friendliness, and efficiency. The object-oriented development approach cannot directly improve all the desirable qualities of software systems. It focuses primarily on improving the ______________________ and ______________________ of software systems.

3. (2 points) A UML ______________________ diagram is for modeling the requirements of a software system. It provides an important link between the requirements and an object-oriented analysis and design model.

4. (2 points) Which of the followings is an incorrect statement about a UML state machine diagram?

   (a) It describes the static structure of a system.
   (b) It depicts the flow of control using states and transitions.
   (c) It is a generalization of the finite state machine.
   (d) It is well suited for modeling a reactive system.
5. (2 points) An object-oriented framework is a semi-complete application, providing the basic structure (or backbone) and utilities for applications to allow one to build an application quickly. And a(n) ____________________ refers to the fact that the top-level control of an application resides in a framework itself, not in the code written by an application developer.

6. (5 points) Describe briefly two different kinds of exceptions: checked exceptions and unchecked exceptions.

7. (5 points) When is a type (a class, an interface, and an array) a subtype of another type in Java?

8. (10 points) What are the two forms of the polymorphism supported by object-oriented programming languages? Describe and compare them briefly.
9. (15 points) Translate the following UML class diagram into Java skeleton code. Your skeleton code should implement all the design decisions expressed in the class diagram; but no need to provide method bodies.
10. (total 20 points) Consider the partial implementations of the Car class given below. In the subquestions below you are to write its `equals` method.

```java
public class Car {
    /** Name of this car. */
    private String name;

    /** Year of this car. */
    private int year;

    /** Make of this car. */
    private Company make;

    /** Create a new car of the given name, year and make. */
    public Car(String name, int year, Company make) {
        this.name = name;
        this.year = year;
        this.make = make;
    }

    /** Overridden here to define an equality of cars. */
    public boolean equals(Object obj) { /* ... */ }

    // rest of code
}

public class Company {

    /** Create a new company of the given name. */
    public Company(String name) { /* ... */ }

    // rest of code
}
```

(a) (5 points) Write the contract of the `equals` method, i.e., properties or constraints that its implementation has to satisfy.
(b) (10 points) Provide the body of the `equals` method. Assume two cars are equal if they are of the same year and have equivalent (equals) names and companies.

(c) (5 points) The above code of the Car class has a problem in that it is missing one important method. What method is missing? Provide its definition. (Hint: it needs to override a method inherited from the class Object.)
11. (10 points) Write a JUnit test class named CarTest to test the equals method of the Car class from the previous question. Try also to ensure the contract of the equals method; that is, provide test cases for testing the contract. You may assume that the Car class defines such access methods as getName(), getYear(), and getMake().
12. (total 15 points) You are to write an applet whose design and partial implementation are given. The applet animates a balloon, a filled circle, positioned at the center of the screen. Initially the balloon is completely deflated (i.e., not visible) and it grows \texttt{delta} pixels in radius at every 100 milliseconds. In sub-questions (a)–(b) below, you are to provide the missing code.

```
import java.awt.*;
import javax.swing.Timer;

public class BalloonApplet extends java.applet.Applet {

    /** Dimension of this applet. */
    protected Dimension dim;

    /** Animation timer. */
    private Timer timer;

    /** Balloon to draw and animate. */
    private Balloon balloon;

    /** Overridden here to create the animation timer and a balloon. */
    public void init() {
        dim = getSize();
        balloon = createBalloon();
        timer = new Timer(100, e -> repaint());
    }

    /** Create a balloon to animate. */
    protected Balloon createBalloon() {
        return new GrowingBalloon(dim.width/2, dim.height/2, Color.GREEN);
    }

    /** Overridden here to draw the balloon. */
    public void paint(Graphics g) {
        balloon.draw(g, dim.width, dim.height);
    }

    /** Overridden here to start the animation timer. */
    public void start() {
        timer.start();
    }

    /** Overridden here to stop the animation timer. */
    public void stop() {
        timer.stop();
    }
}
```
(a) (5 points) Write the interface Balloon (Hint: how is the interface used by the BalloonApplet class?)

```java
import java.awt.*;
public interface Balloon {
}
```

(b) (10 points) Write the class GrowingBalloon that implements the interface Balloon by completing the following skeleton code. (Hint: to draw a balloon, use the method void fillOval(int x, int y, int width, int height) of the Graphics class.)

```java
import java.awt.*;
public class GrowingBalloon /* YOUR CODE HERE */ {
    /** Coordinates of this balloon. */
    protected int x, y;

    /** Current radius of this balloon. */
    protected int radius = 0;

    /** Growing rate of this balloon’s radius. The radius grows * delta pixels every time the draw method is called. */
    protected int delta = 2;

    /** Color of this balloon. */
    protected Color color;

    /** Create a new balloon at the given position and with the given color. */
    public GrowingBalloon(int x, int y, Color color) {
        this.x = x;
        this.y = y;
        this.color = color;
    }

    /** Increases the balloon’s radius by delta pixels and draws it using * the given the graphics. */
    public void draw( // WRITE YOUR CODE HERE INCLUDING PARAMETERS... }
13. (10 points) Write a subclass of the `GrowingBalloon` class, named `GrowingShrinkingBalloon`, by completing the partial code given below. This refined balloon not only grows but also shrinks. As before, the balloon continues to grow delta pixels in radius each time the `draw` method is called, but if the balloon reaches or touches any of the four sides of the screen, it starts to shrink at the same rate as the growing rate. The shrinking balloon starts to grow again when it becomes completely deflated—i.e., when the radius becomes 0. Thus, the balloon repeats the growing and shrinking phases. Write minimal code by reusing as much superclass code as possible. (Hint: use the static method `Math.min(int, int)` to determine the minimum of two values. If your code is more than 5 lines long, you are doing it in a hard way.)

```java
import java.awt.*;
public class GrowingShrinkingBalloon extends GrowingBalloon {

    /**
     * Create a new balloon at the given position and with the given
     * color. The initial size of the balloon is 0.
     * /
    public GrowingShrinkingBalloon(int x, int y, Color color) {
        super(x, y, color);
    }

    /** Adjusts this balloon's radius by delta pixels and draws it. */
    public void draw( // WRITE YOUR CODE HERE INCLUDING PARAMETERS...