Consider the following class definitions. Assume that each class is defined in its own .java file.

```java
public abstract class Cake {
    protected boolean frosting; // field
    public abstract String recipe(); // abstract method
    public String toString() { return "Something sweet!"; }
}

public class ChocolateCake extends Cake {
    public ChocolateCake (boolean addFrosting) { this.frosting = addFrosting; }
    public String recipe() { return "Prepare the dough and bake."; }
    public String toString() {
        if (this.frosting){ return super.toString()+ " with frosting and chocolate";)
        else{ return super.toString()+ " with chocolate"; }
    }
}

public class BirthDayChocoCake extends ChocolateCake {
    protected int nCandles; // additional field
    public BirthDayChocoCake (boolean addF, int numCandles) {
        super(addF);
        this.nCandles = numCandles;
    }
    public void putCandles(){ System.out.println("The cake has "+ nCandles +" candles");}
}
```

1. (2 pts) Put an X in the right answer cell (compilation error, runtime error or NO error).

<table>
<thead>
<tr>
<th></th>
<th>Compilation Error</th>
<th>Runtime Error</th>
<th>No error</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChocolateCake c3 = new ChocolateCake( true ); ((BirthDayChocoCake) c3).putCandles();</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cake c4 = new BirthDayChocoCake( false, 5 ); String message = c4.recipe();</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>BirthDayChocoCake c = new ChocolateCake(true);</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cake c2 = new Cake( false );</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. (3 pts) In the box, write what the following code will print when main is called (there are no errors in this code):

```java
class MyTester {
    public static void main( String[] args ) {
        Cake c1 = new ChocolateCake( true );
        Cake c2 = new BirthDayChocoCake( false, 3 );
        ( (BirthDayChocoCake) c2).putCandles();
        System.out.println( c2 );
        System.out.println( c1 );
    }
}
```
3. (2 pts) Consider the following code:

```java
import javax.swing.*;
import java.awt.*;
public class MyPanel extends JPanel {
    protected void paintComponent( Graphics g ) {
        super.paintComponent(g);
        g.drawLine(10, 10, 100, 100);
        g.drawLine(100, 100, 190, 10);
    }

    public static void main( String[] args ) {
        JFrame f = new JFrame();
        f.setSize( 200, 200 );
        JPanel p = new MyPanel();
        f.add( p );
        f.setVisible( true );
    }
}
```

CIRCLE the option that applies (True or False) for each sentence:

True  False  This code will display the second window (the empty one), because only the JFrame is visible.

True  False  At runtime, Java will detect that the object referenced by `p` is actually a `MyPanel` object and will use the `paintComponent` method in the `MyPanel` class.

True  False  If we remove the line `f.add( p );` the `JPanel` `p` won’t be added to our `JFrame` `f` and therefore we won’t see the `V` drawing in the window displayed after running the main method.

True  False  The line `super.paintComponent(g)` included in `MyPanel`'s `paintComponent` method tells Java that this method overrides the `paintComponent` method from `JPanel`. i.e., without that line, in this code Java would use the `paintComponent` method from the `JPanel` class instead of the one from the `MyPanel` class.

4. (3 pt) Fill the gaps (ONLY ONE WORD in each GAP).

In the sample code from previous question, if we want to assign something to the variable `p`, we can assign a reference to an object of the same class than `p`, or to an object of any ____________ of JPanel and it will be correct; this is know as ____________.

___________ enables you to define a general class (e.g., a superclass) and later extend it to more specialized classes (e.g., subclasses).

In the code from previous question, `MyPanel` is a ____________ of `JPanel`, and `JPanel` is the ____________ of `MyPanel`. This implies that the class ____________ inherits fields and methods from the class `JPanel`.

Every class in Java has a common superclass, the ____________ class.