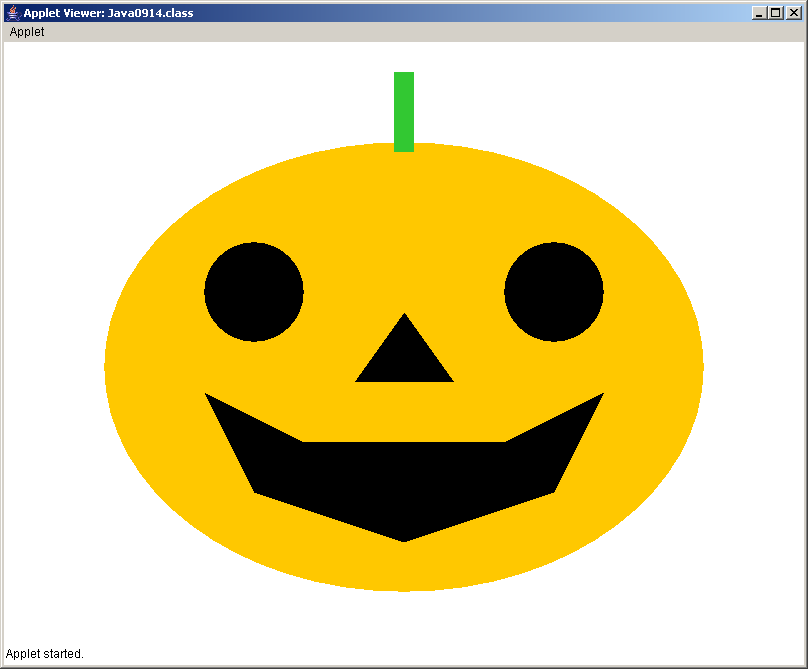
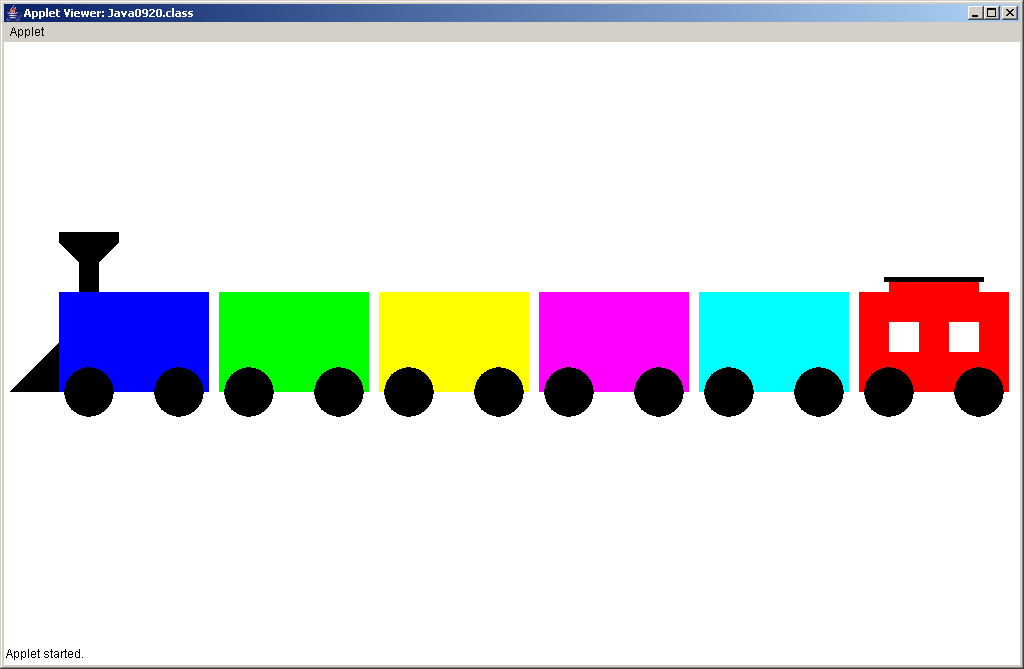
|  |  |
| --- | --- |
| **AP Computer Science** | **GraphicsLab04 Java Assignment** |
| **The Composition/Inheritance**  **Graphics Program** | **80, 90, 100 & 110 Point Versions** |
| **Assignment Purpose:**  This assignment has multiple goals. You need to demonstrate your knowledge of composition and inheritance, demonstrate creativity and demonstrate the ability to work on this assignment with a partner in a small team. | |

Write an **applet** program that demonstrates composition and inheritance. These concepts need to be shown in graphics format similar to the chapter 9 **Jack-O’-lantern** and **Train** case studies. In the case of a **Jack-O’-lantern**, a **Jack-O’-lantern** *is-a* **Pumpkin**, and a **Jack-O’-lantern** *has-a* **Face**. In the case of a **Train**, a **Train** *has-a* **Locomotive**, **TrainCars**, and a **Caboose**; however, a **Locomotive** *is-a* **TrainCar** and a **Caboose** *is-a* **TrainCar**.

This is an **OPEN-ENDED** lab assignment. Frankly, you will find this harder than your previous lab assignments. The required **main** method is not shown. The required program executions for different point versions are not available. This is an assignment written with the creativity of your team. Even though you have control over the classes that you select and the graphics images that you display, there are point versions that need to be satisfying certain requirements.

**80 Point Version**

Your program needs to demonstrate one example of inheritance and one example of composition. This will mean a minimum of three separate classes. You must be aware that the Java syntax has to follow the rules of composition and inheritance. In other words you cannot simply declare some classes that have the required properties in real life. For instance an **Animal** *has-a* **Head** and a **Tiger** *is-an* **Animal**. That is fine, but the Java program statements must correctly use the relationships.

A good example of an 80-point program is the **Jack-O’-lantern** class. This is a simple program that demonstrates inheritance and composition, but does not pass any parameters other than the **Graphics** object **g**.

**90 Point Version**

This version required some other parameter, besides **Graphics** object **g**, is passed to the *subclass*. It also requires that this information is used by the subclass in a meaningful fashion. A good example of this is the **TrainCar** class in program **Java0917.java**. If this is done correctly you should be able to create multiple objects that are noticeably different by providing different parameters.

**100 Point Version**

There are 3 ways to earn a 100.

***Method 1***

One way to earn a 100 is to demonstrate *multi-level inheritance*.

For example: a **Eagle** *is-a* **Bird** and a **Bird** *is-an* **Animal**.

***Method 2***

Another way to earn a 100 is to pass one or more parameters to the *SUPERCLASS*.

This requires that the information is first passed to the *subclass* constructor.

Then the information needs to be redirected to the *superclass* constructor with the **super** keyword.

It also requires that this information is used by the *superclass* in a meaningful fashion.

A good example of a 100-point program is the **Train** class.

***Method 3***

Do the 90 point version, and earn “Creativity Points”. Explain below.

**Extra Creativity Points**

Ten points can be earned for a total possible of **110 points** for creativity and appearance above and beyond the program requirements. The program needs to correctly show composition and inheritance and these relationships must be shown graphically. However, there is no graphics requirement beyond correctness. If the graphics display is especially creative and original you can earn these 10 points, which **please realize** are subjective points.