|  |  |
| --- | --- |
| **AP Computer Science GridWorld Case Study (GWCS)** | |
| **GridWorldLab06** | **Creating New Classes with Inheritance** |

|  |
| --- |
| **Lab Objectives** |
| Create a new class by using inheritance | |

|  |
| --- |
| **Lab Prerequisites** |
| **Completed ExpoJava, Chapter 09 and completed GridWorldLab05** | |

|  |  |
| --- | --- |
| **Lab Sequence of Steps** | |
| **#** | **Actions** | | **Comments** |
| **01a** | **Check the Files for GridWorldLab06**  Open **My Documents** and navigate to the **GridWorldLab06** folder.  You will get the display with 2 folders, like **Figure 01**.  Each folder contains files for a different project.  **Figure 01** | | This lab assignment contains the files for 2 completely different projects. One is for the 80 point version, and the other is for the 100. |
| **02a** | **80 Point Version Requirements**  Create a new file called **CompassBug.java** in the appropriate folder.  A **CompassBug** *is-a* **BoxBug** whose color changes according to which direction it is going.  North - Red  East - Green  South - Blue  West - Black  No changes are to be made to the **BoxBug** superclass.  All work is done in the **CompassBug** subclass.  When the project is created, compiled and executed, you should see the output in **Figure 02**.  **Figure 02** | | Leaving the superclass file intact is one way inheritance helps to ensure reliability. |

|  |  |  |
| --- | --- | --- |
| **02b** | **100 Point Version Requirements**  Create a new file called **Frog.java** in the appropriate folder.  A **Frog** *is-a* **Bug** that moves 2 cells at a time – which means it can jump over other objects than get in its way as long as there is an empty cell on the other side for it to land. When a **Frog** cannot **move**, it will **turn**.  When the project is created, compiled and executed, you should see the output in **Figure 03**.  **Figure 03** | At this point of the execution, notice that the **Frog** on the left has a **Rock** directly in front of it, while the other **Frog** has an empty cell in front of it.  In the very next **step** of the execution, we see that the **Frog** on the left jumped over the **Rock** while the other **Frog** had to **turn**. Even though there was an empty cell in front of the **Frog**, it had to turn because it would have *landed* on a **Rock**. |