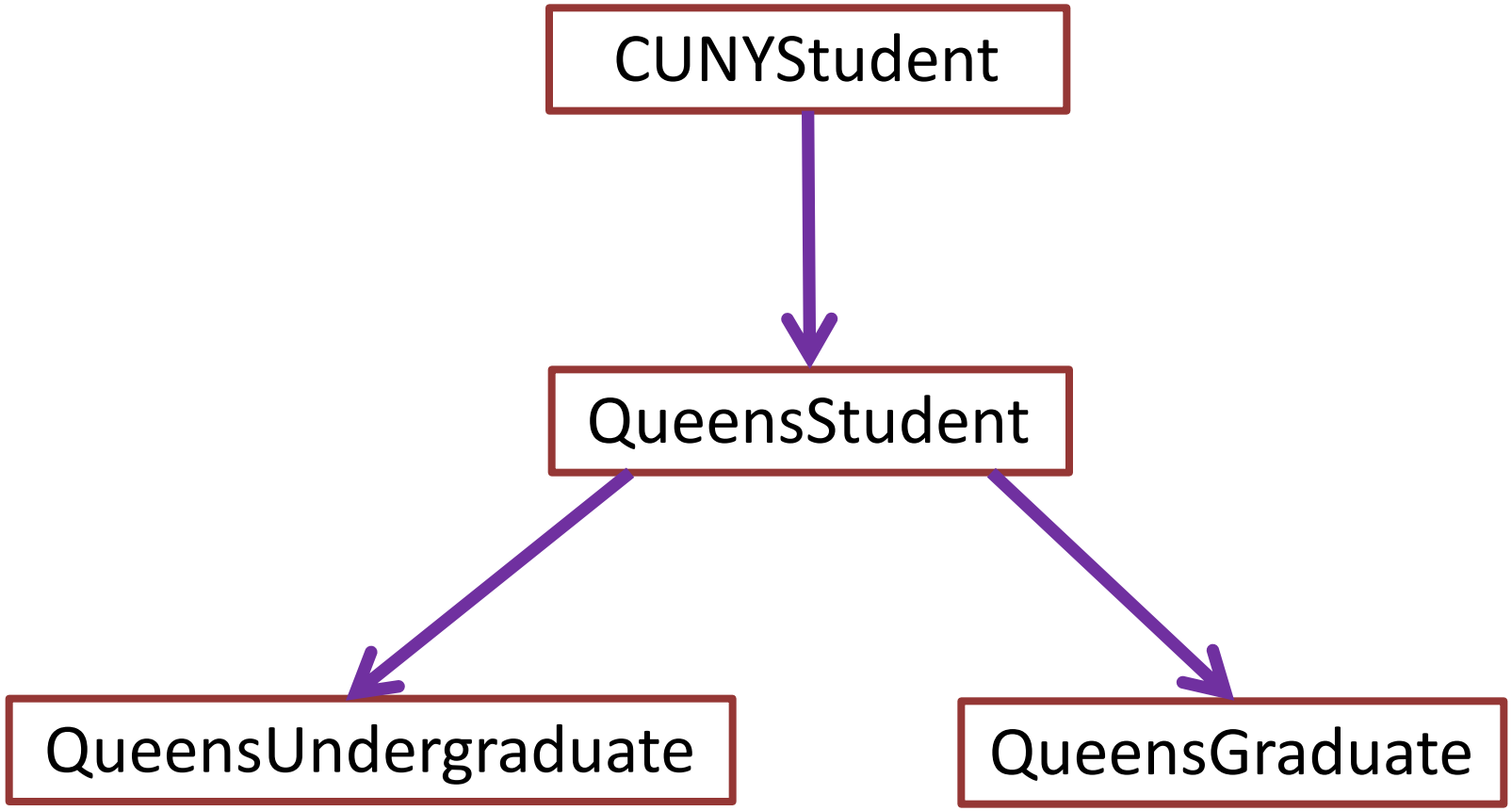
Inheritance, Polymorphism and Abstract Classes

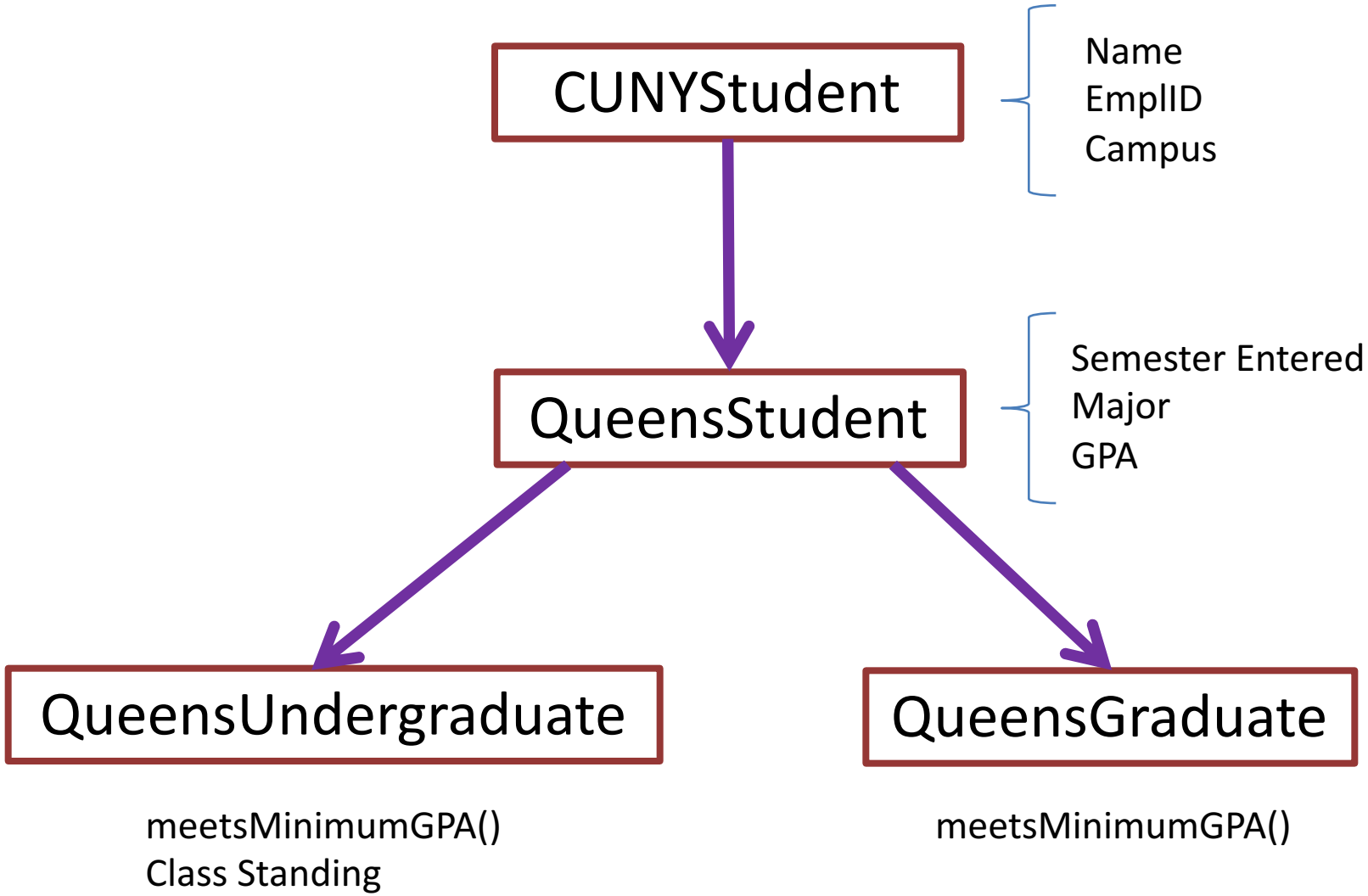
Student Management System

- All students are CUNY Students
- CUNY Students are Queens College students, or Hunter College students, or City College…
- Queens College students may be undergraduate or graduate students.
- There is a different minimum GPA for undergraduates and graduate students

Create a Class Hierarchy

- Extend classes based on the "*is a*" relationship
- A **Cat** *is a* **Pet** so class Cat extends Pet
- An SSNGUI *is a* Jframe so SSNGUI extends Jframe
- A Queens College Student *is a* CUNY Student so Queens College Student extends CUNY Student.
- etc.


CUNYStudent
QueensStudent
QueensUndergraduate
QueensGraduate


CUNYStudent
Name
EmplID
Campus
QueensStudent
Semester Entered
Major
GPA
QueensUndergraduate
meetsMinimumGPA()
Class Standing
QueensGraduate
meetsMinimumGPA()

```
public class CUNYStudent {
   private String emplID;
   private String name;
   private String campus;
}

public class QueensStudent extends CUNYStudent {
   private String semesterEntered;
   private String major;
   private float gpa;
}

public class QueensUndergraduate extends QueensStudent {
   public boolean meetsMinimumGPA() {
      return gpa >= 2.0f;
   }
}

public class QueensUndergraduate extends QueensStudent {
   public boolean meetsMinimumGPA() {
      return gpa >= 3.0f
   }
}
```

```
public class CUNYStudent {
   private String emplID;
   private String name;
   private String campus;
}

public class QueensStudent extends CUNYStudent {
   private String semesterEntered;
   private String major;
   private float gpa;
}

public class QueensUndergraduate extends QueensStudent {
   public boolean meetsMinimumGPA() {
      return gpa >= 2.0f;
   }
}

public class QueensUndergraduate extends QueensStudent {
   public boolean meetsMinimumGPA() {
      return gpa >= 3.0f
   }
}
```

The *protected* modifier

The *protected* modifier grants access only from descendant classes

Pubic grants access from *any* class.

Private grants access only to instances of the same class.

```
class X {
        public int a;
        protected int b;
        private int c;
        a = 1;
        b = 2;
        c = 3;
```

```
class Y extends X {
        a = 1;
        b = 2;
        c = 3;
```

```java
public class CUNYStudent {
   private String emplID;
   private String name;
   private String campus;
}

public class QueensStudent extends CUNYStudent {
   private String semesterEntered;
   private String major;
   protected float gpa;
}

public class QueensUndergraduate extends QueensStudent {
   public boolean meetsMinimumGPA() {
      return gpa >= 2.0f;
   }
}

public class QueensUndergraduate extends QueensStudent {
   public boolean meetsMinimumGPA() {
      return gpa >= 3.0f
   }
}
```

Constructors

When a class is instantiated the *first thing* it must do is "construct" its super class.

Calling one of the constructors of the super class is done using the method

super(<optional parameters>)

```
public class CUNYStudent {
   private String emplID;
   private String name;
   private String campus;

   public CUNYStudent (String theCampus) {
      campus = theCampus;
   }

   public CUNYStudent (String theEmplID,
                       String theName,
                       String theCampus) {
      emplID = theEmplID;
      name = theName;
      campus = theCampus;
   }

}
```

The name of the constructor is the same as the name of the class

The constructor has no return type.

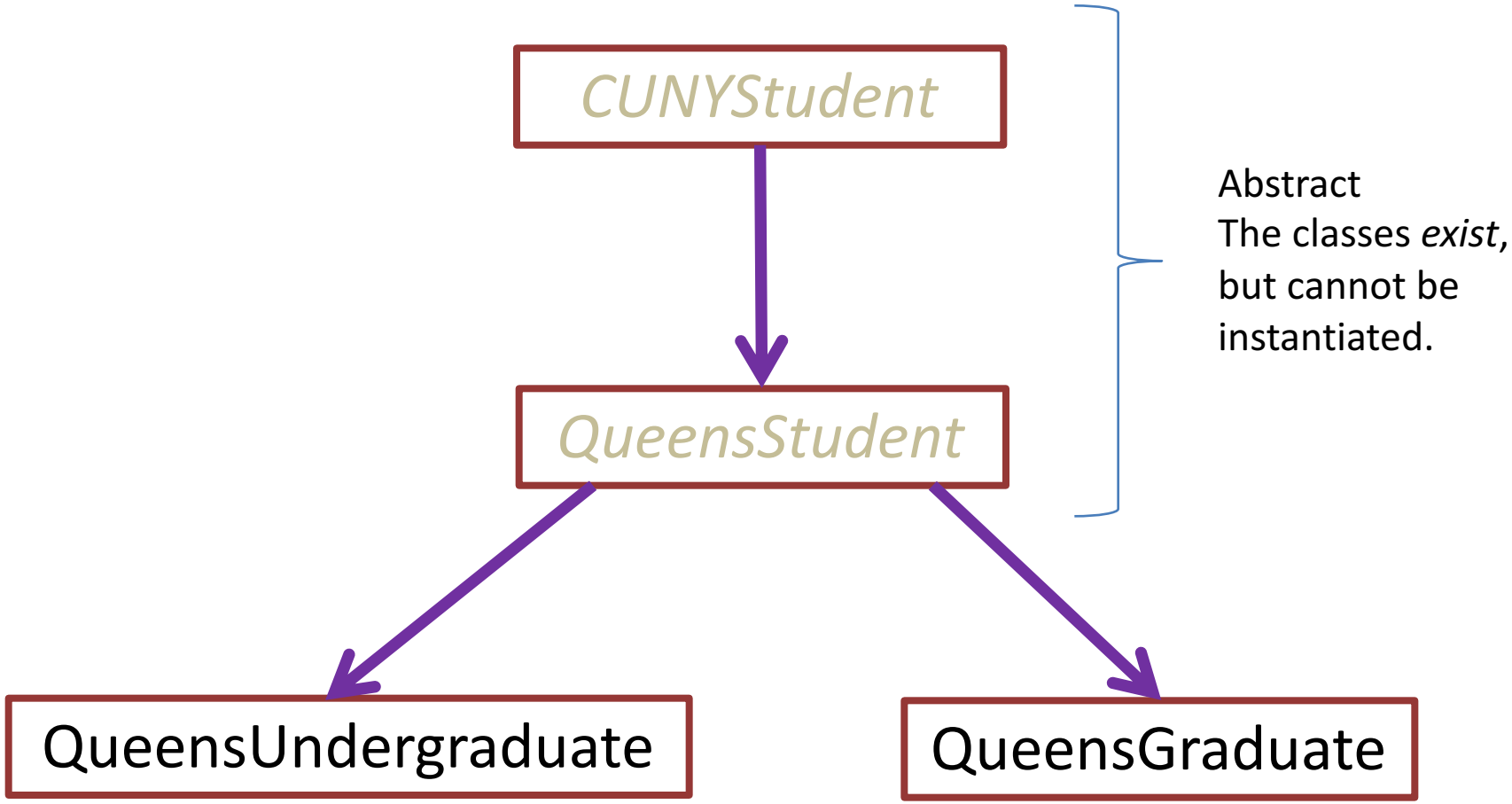
```java
public class QueensStudent extends CUNYStudent {
   private String semesterEntered;
   private String major;
   protected float gpa;

   public QueensStudent () {
      super ("Queens");
   }

   public QueensStudent (String theSemester,
                         String theMajor,
                         float theGPA) {
      super ("Queens");
      semesterEntered = theSemester;
      major = theMajor;
      gpa = theGPA;
   }
}
```

```java
public class QueensUndergraduate extends QueensStudent {
   private int classStanding;

   public QueensUndergraduate (String theSemester,
                               String theMajor,
                               float theGPA,
                               int standing) {
      super (theSemester, theMajor, theGPA);
      classStanding = standing;

   }
   public boolean meetsMinimumGPA() {
      return gpa >= 2.0f;
   }
}
```

```
public class Demo1 {

      public static void main (String[] args) {
         CUNYStudent[] students;
         students = new CUNYStudent[10];
         students[0] = new QueensUndergraduate("1132","CSCI-BA",3.23f,4);
         students[1] = new QueensUndergraduate("1129","CSCI-BS",2.14f,4);
         students[2] = new QueensGraduate("1132","MUSIC-BA",3.55f);
         students[3] = new QueensUndergraduate("1139","ACCT-BA",3.56f,4);

         listPassing(students,4);

      }
      public static void listPassing(CUNYStudent[] studentList, int numFilled) {
         for (int i=0; i<numFilled; i++) {
            if (studentList[i].meetsMinimumGPA())
               System.out.println(studentList[i]);
         }
      }
}
```

Abstract Classes

- An *Abstract Class* cannot be instantiated
- A class is abstract if
  - It is declared as abstract
  - It contains an abstract method
  - It inherits an abstract method and does not overload it.


CUNYStudent
QueensStudent
QueensUndergraduate
QueensGraduate
Abstract
The classes *exist*,
but cannot be
instantiated.

```java
public abstract class QueensStudent extends CUNYStudent
{
   private String semesterEntered;
   private String major;
   protected float gpa;

   public QueensStudent () {
      super ("Queens");
   }

   public QueensStudent (String theSemester,
                         String theMajor,
                         float theGPA) {
      super ("Queens");
      semesterEntered = theSemester;
      major = theMajor;
      gpa = theGPA;
   }

}
```

The *instanceof* Operator

```
public static void whatKindOfStudent
              (CUNYStudent[] studentList, int numFilled) {
         String kindOfStudent = null;

   for (int i=0; i<numFilled; i++) {
      if (studentList[i] instanceof QueensUndergraduate)
         kindOfStudent = "an undergraduate";
      if (studentList[i] instanceof QueensGraduate)
         kindOfStudent = "a graduate";

      System.out.println(studentList[i]+" is " +
                         kindOfStudent);
         }
      }
```