Computer Science AP
Polymorphism Worksheet

These are questions associated with each video in the Polymorphism Section.

Questions for Video: Introduction to Polymorphism 01

Use the project InheritanceExamples for these questions.

Consider the StudentTeacher01 package.
State *True* or *False* as to whether or not the following lines would cause a compile time error.
I've written error in front of the ones that cause errors!

```
      Student S=new Student("Adam", 15, 123)
Error Student S=new Person("Adam", 15)
      Person P=new ExchangeStudent("Adam", 15, 123)
Error ExchangeStudent S=new Student("Adam", 15, 123)
      Person P=new ExchangeStudent("Adam", 15, 123)
Error ExchangeStudent S=new Person("Adam", 15)
      Person P=new Teacher("John", 45)
Error Teacher T=new Person("John", 45)
```

Consider the LCDScreen package.
State *True* or *False* as to whether or not the following lines would cause a compile time error.

```
      Screen S=new ColorScreen("a83jf")
Error ColorScreen S=new Screen("fji8d")
      ColorScreen S=new ColorScreenPlus("sif8d", 123)
Error ColorScreenPlus S=new Screen("ajf8d")
      Screen S=new ColorScreenPlus("ji8de", 123)
```

A student creates a project that consists of exactly three classes. The three classes make use of inheritance through the use of 'extends'. The following lines compile without error. Can you determine which classes extend which classes? Explain.

```
Goobly G = new FaFoo()
Babber B = new FaFoo()
Goobly G = new Babber()
```

Goobly
Babber extends Goobly
Fafoo extends Babber

Questions for Video: Introduction to Polymorphism 01

There were several rules about what the compiler will allow when it comes to reference fields calling methods on instances that may or may not be subclasses of the reference field type (example: Student S = new ExchangeStudent("Adam", 15, 123) The instance in memory is not of the same type as the field type Student). Use the InheritanceExamples01 project to answer the following questions to check whether or not you have a good understanding about these rules! Remember there is a logic as to WHY the rule is in place. Understand the logic and the rule should be easier to learn.

Use the StudentTeacher01 package for these questions.

Person P=new Student("Adam", 15,123)
Which lines will cause a compile time error?
___ P.talk()
___ P.setAge(15)

Error P.study()
P is a Person reference and Person cannot study.
Error double g=P.getAverageMark()
P is a Person rerefence and Person cannot getAverageMark.
___ P.stuff()

Student S=new ExchangeStudent("Adam", 15, 123)
Which lines will cause a compile time erorr?
___ S.talk()
___ S.stuff()
___ double d=S.getAverageMark()
Error double d=S.getUnadjustedAverageMark()
S is a Student reference and Student doesn't have getUnadjustedMark
Error String s=S.getReturnDate()
S is a Student reference and Student doesn't have getReturnDate

Consider the concept in polymorphism that you can cast instances in memory into other types if required and when acceptable. Which of the following are acceptable casts?

Pesron P=new ExchangeStudent("Adam", 15, 123) ((ExchangeStudent)P).getReturnDate()	good. the code to be an exchangestudent is there in memory.
Person P=new ExchangeStudent("Adam", 15, 123) ((Student)P).study()	good. the code to be a student is there in memory.
Person P=new Student("Adam", 15, 123) ((ExchangeStudent)P).getReturnDate()	NO. only the code for student is in memory so can't be turned into an exchangestudent
Student S=new Student("Adam", 15, 123) ((Person)S).stuff()	good. the code to be a person is there in memory.
Student S=new Student("Adam", 15, 123) ((ExchangeStudent)S).getReturnDate()	NO. only the code for student is in memory so can't be turned into an exchangestudent

Person P = new Teacher(“John”, 45) ((Teacher)P).greet(Adam) //assume Adam is a Person...	good. P references a Teacher, so it can be cast as a teacher.

Indicate whether the following code segments are good, have compile time errors (detected by the IDE), or run time errors (will cause error when running). For segments with errors you should clearly indicate the reason for the error.
Use the LCDScreen package for code reference.

*remember, casting errors are run time
*remember, a reference type can only call methods that it has (doesn’t matter what is actually stored in memory) , this is what the compiler checks

Screen S=new ColorScreen(“DC5”) S.display(“Hello”)	good. Screens can call display. COMPILE TIME ERROR
Screen S=new ColorScreen(“DC5”) S.setColor(Color.BLACK) //Color.Black is an acceptable Color...	bad. S is a screen reference. It cannot call setColor. COMPILE TIME ERROR
Screen S=new ColorScreen(“DC5”) ((ColorScreenPlus)S).useSpecialEffect(3)	bad. RUNTIME ERROR S is a colorscreen in memory and cannot be cast into a colorscreenplus since there are extra methods needed.
Screen S=new ColorScreen(“DC5”) ((ColorScreen)S).useSpecialEffect(3)	bad. The cast will work BUT a colorscreen cannot use that method! COMPILE TIME ERORR
ColorScreen CS=new ColorScreen(“DC5”) CS.setVertical(150)	bad. ColorScreen doen’t have that method. COMPILE TIME ERROR

Questions for Video: Polymorphism in Action 01 and 02

Since a reference type can possibly reference more than one type of object in memory, more possibilities open up when it comes to method arguments and method return types. Here are some questions to check if you picked up the basics on this topic.

Use the InhertianceExamples project, StudentTeacher01 package for these questions.

Teacher T=new Teacher("John", 45)
T.greet (???)
T.makeStudy(???)

What type/s of instances from this project could you pass as an argument into the Teacher's greet method?
Since the parameter is type Person, you could send in a Person, Student, ExchangeStudent. Anything that qualifies as a Person.

What type/s of instances from this project could you pass as an argument into the Teacher's makeStudy method?
Student or ExchangeStudent. Anything that qualifies as a Student.

Student Adam=new Student("Adam", 15, 123)
ExchangeStudent Bob=new ExchangeStudent("Bob", 15, 456)
Adam.partnerUp(Bob)

Will partnerUp method accept Bob as an argument? Explain.
Yes. Parameter is type Student. This means that any class that qualifies as a Student can be sent in to the method.

A runner program uses the following lines that use the Teacher class' pickStudent method. Will the following lines compile and run? Explain. Assume that T is a teacher.

Student volunteer = T.pickStudent()
ExchangeStudent someone = T.pickStudent()
Person temp = T.pickStudent()

1st line, all good
2nd line, no. The pick Student method send back a Student type. This isn't good enough! It needs to send back an ExchangeStudent.
3rd line, all good. Any type of student qualifies as a person.

Assume that we add a method to the Student class has a method with the signature
public Person getFavoritePerson()
Could this method return an instance of type Student? ExchangeStudent? Teacher? Explain.
Yes. All these types qualify as a person since they extend off of person.

Use the InhertianceExamples project, LCDScreen package for these questions.

A runner program has an ArrayList of type Screen. The list contains several items.
ArrayList<Screen> screens = new ArrayList<Screen>()

What type/s of instances can be placed in this list?
All the types of screens could be added into the list. They all extend off of Screen.

For this question assume that all the screens in the list are either ColorScreens or ColorScreenPlus. Would the following line work in the program? Explain.
ColorScreen temp = screens.get(0)

NO. The list stores type Screen, so get(0) sends back a reference type to a screen. You can't set a ColorScreen to a Screen type. Even though YOU KNOW that the actual instance in memory is a colorScreen or colorScreenPlus the compiler doesn't consider what is in memory. It just cares about making sure types match. It would have to be
Screen S = screens.get(0)
Then you could cast S as a colorscreen or colorscreenplus if you knew it was that type.

What code would grab the very first item in the list and ask it to display "Hello!" ?

```
Screen S = screens.get(0);
S.display("Hello");
```

Assume that you don't know whether the first item in the list is a Screen, ColorScreen, or ColorScreenPlus. Will the line of code you wrote for the question above work regardless of the screen type? Explain why or why not.
Yes it will work. All the screen types inherit the display method from Screen class.

What is the role of the operator 'instanceof'?
Can check if an instance is of a certain class or not. Subclasses count as a class type.

What code would grab the very first item in the list, check to see if it is a ColorScreenPlus, and if it is a ColorScreenPlus ask it to use special effect code 5.

```
Screen S = screens.get(0);
if (S instanceof ColorScreenPlus) {
   ( (ColorScreenPlus)S).useSpecialEffects(5) );
}
```