Exercises About Inheritance and Polymorphism in Java

Dr. Huda Hakami

Question 1

Which statement about abstract classes and interfaces is *false*?

(A) An interface cannot implement any methods, whereas an abstract class can.
(B) A class can implement many interfaces but can have only one superclass.
(C) An unlimited number of unrelated classes can implement the same interface.
(D) It is not possible to construct either an abstract class object or an interface object.
(E) All of the methods in both an abstract class and an interface are public.

Question 1

Which statement about abstract classes and interfaces is *false*?

(A) An interface cannot implement any methods, whereas an abstract class can.
(B) A class can implement many interfaces but can have only one superclass.
(C) An unlimited number of unrelated classes can implement the same interface.
(D) It is not possible to construct either an abstract class object or an interface object.
(E) All of the methods in both an abstract class and an interface are public.

Answer: E

Question 2

Which statement about interfaces is true?

I An interface contains only public abstract methods and public static final fields.
II If a class implements an interface and then fails to implement any methods in that interface, then the class *must* be declared abstract.
III While a class may implement just one interface, it may extend more than one class.

(A) I only
(B) I and II only
(C) I and III only
(D) II and III only
(E) I, II, and III

Question 2

Which statement about interfaces is true?

I An interface contains only public abstract methods and public static final fields.
II If a class implements an interface and then fails to implement any methods in that interface, then the class *must* be declared abstract.
III While a class may implement just one interface, it may extend more than one class.

(A) I only
(B) I and II only
(C) I and III only
(D) II and III only
(E) I, II, and III

Answer: B

Question 3

In Java, methods declared as private can never be overridden.

(A) True

(B) False

Question 3

In Java, methods declared as private can never be overridden.

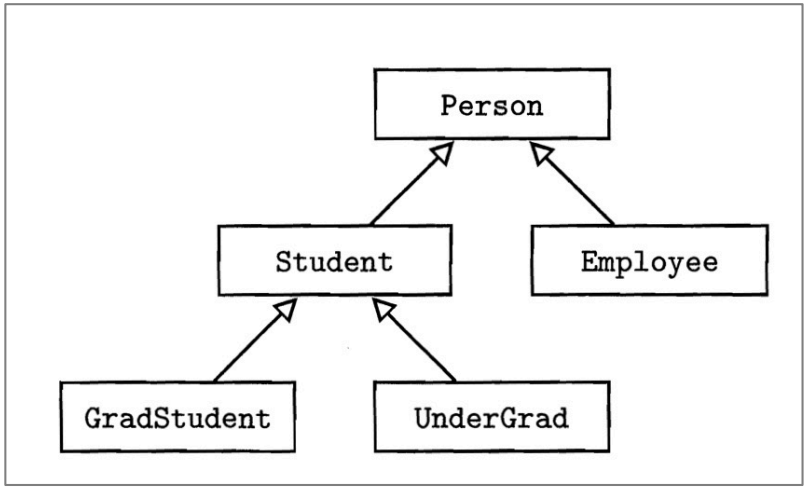
(A) True

(B) False

Answer: A

Question 4

Consider the following class hierarchy:

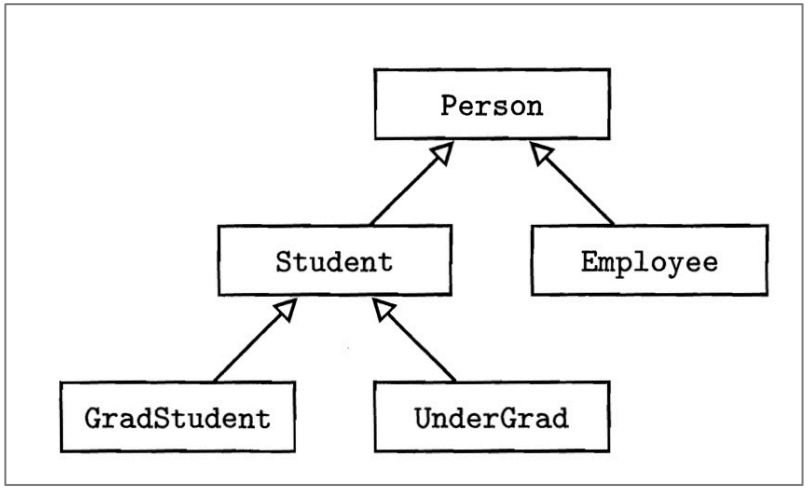


Is each of the following is legal? Why?

```
Student s = new Student();
Student g = new GradStudent();
Student u = new UnderGrad();
```

Question 4

Consider the following class hierarchy:



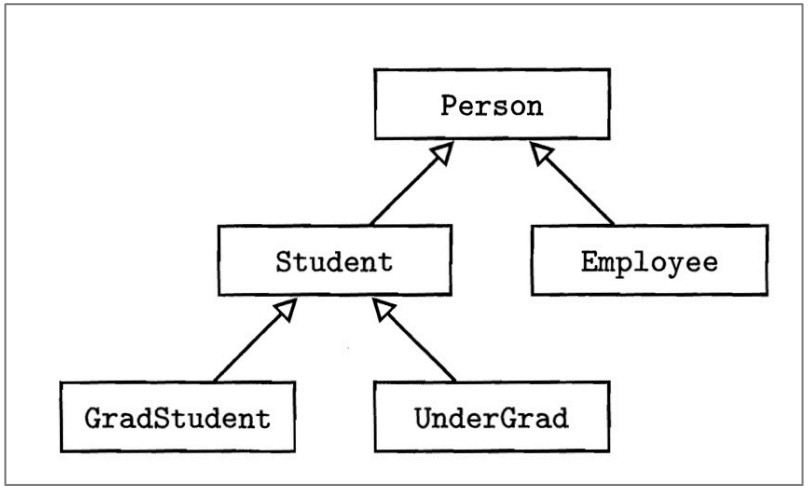
Is each of the following is legal? Why?

```
Student s = new Student();
Student g = new GradStudent();
Student u = new UnderGrad();
```

This works because a GradStudent *is-a* Student, and an UnderGrad *is-a* Student

Question 5

Consider the following class hierarchy:

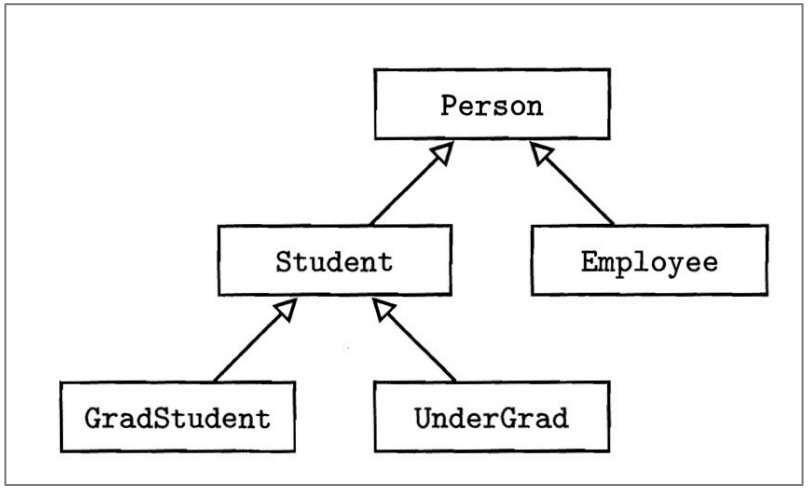


Is each of the following is legal? Why?

```
GradStudent g = new Student();
UnderGrad u = new Student();
```

Question 5

Consider the following class hierarchy:



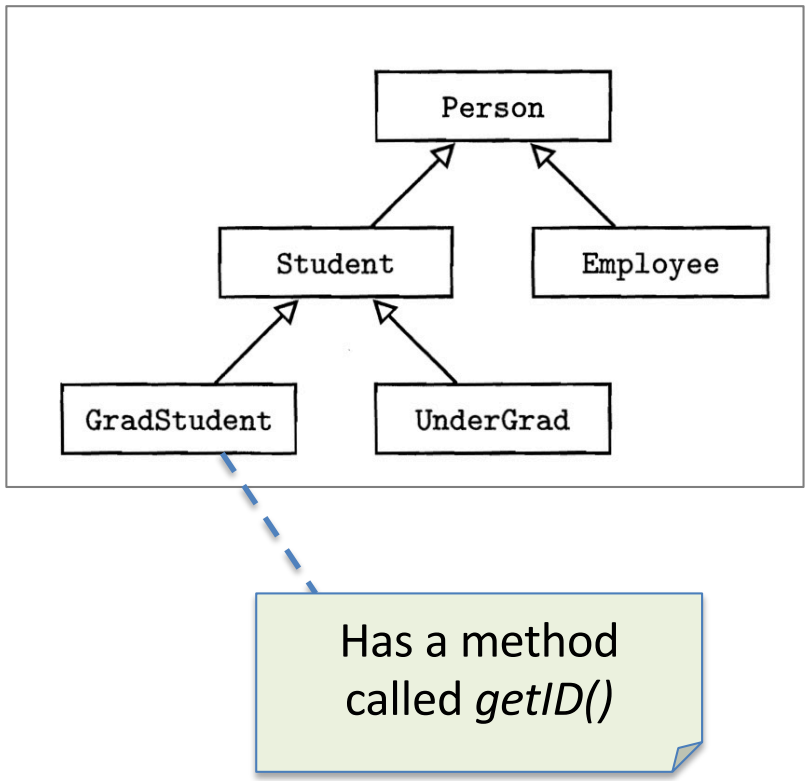
Is each of the following is legal? Why?

```
GradStudent g = new Student();
UnderGrad u = new Student();
```

The previous declarations are not valid, since a Student is not necessarily a GradStudent nor an UnderGrad

Question 6

Consider the following class hierarchy:

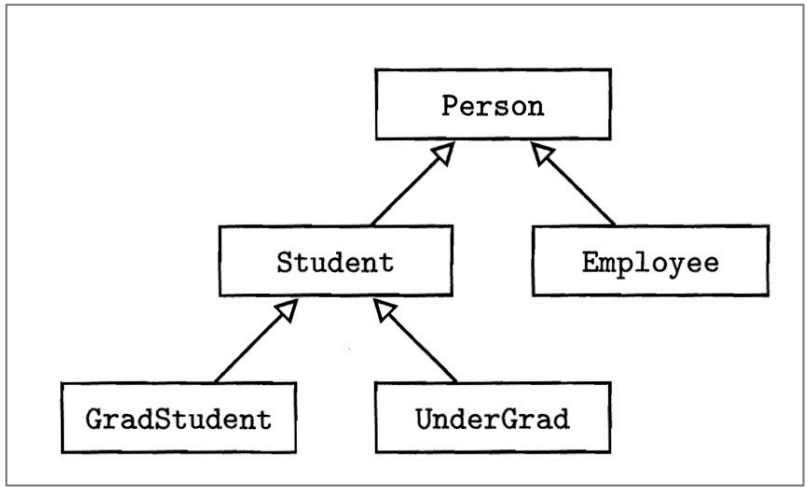


Is each of the following is legal? Why?

```
Student s = new GradStudent();
GradStudent g = new GradStudent()
int x = s.getID();                    //
int y = g.getID();                    //
```

Question 6

Consider the following class hierarchy:



Is each of the following is legal? Why?

```
Student s = new GradStudent();
GradStudent g = new GradStudent()
int x = s.getID();                //
int y = g.getID();                //
```

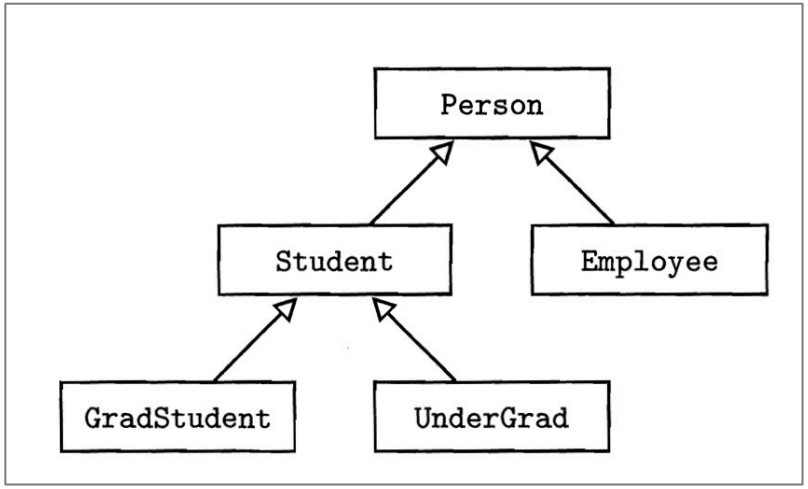
s.getID(); is a compilation error because it invokes a subclass-only method directly on a superclass reference.

In other words:

s is of type Student, and the Student class doesn't have a *getID* method.

Question 7

Consider the following class hierarchy:



Is each of the following is legal? Why?

```
Student s = new GradStudent();
GradStudent g = new GradStudent()
int x = s.getID();                    //
int y = g.getID();                    //
```

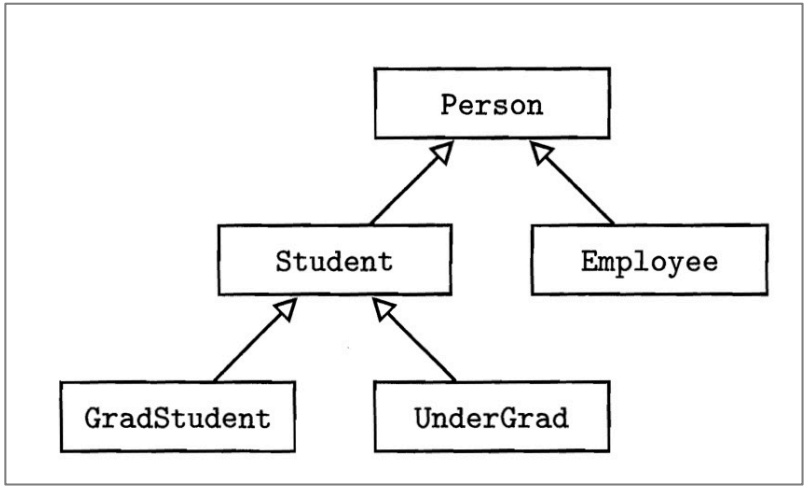
The error can be fixed by **downcasting** s to the correct type as follows:

int x= ((GradStudent) s).getID();

Since *s* (of type Student*)* is actually representing a GradStudent object, such a cast can be carried out.

Question 7

Consider the following class hierarchy:



Is each of the following is legal? Why?

```
Student s = new GradStudent();
GradStudent g = new GradStudent()
int x = s.getID();                //
int y = g.getID();                //
```

g.getID() is **legal** call

Because *g* is declared to be of type GradStudent, and GradStudent class contains getID() method.

Question 8

Which of these rules is not true for subclasses:

- A subclass can add new private instance variables.
- A subclass can override inherited methods.
- A subclass cannot access a the private members of its superclass.
- A subclass may override static methods of the superclass.

About Polymorphism

- Polymorphism is the mechanism of selecting the appropriate method for a particular object in a class hierarchy. The selection of the correct methods occurs during :
  - The run of the program (dynamic binding/late)
  - The compile of the program (static binding/early)

Question 8

Which of these rules is not true for subclasses:

- A subclass can add new private instance variables.
- A subclass can override inherited methods.
- A subclass cannot access a the private members of its superclass.
- A subclass may override static methods of the superclass.

About Polymorphism

- Polymorphism is the mechanism of selecting the appropriate method for a particular object in a class hierarchy. The selection of the correct methods occurs during :
  - The run of the program (dynamic binding/late)
  - The compile of the program (static binding/early)

Question 9

What is downcasting?

a) Casting subtype to supertype
b) Casting supertype to subtype
c) Casting subtype to supertype and vice versa
d) Casting anytype to any other type

Question 9

What is downcasting?

a) Casting subtype to supertype
b) Casting supertype to subtype
c) Casting subtype to supertype and vice versa
d) Casting anytype to any other type

Explanation: The downcasting concept includes only the casting of supertypes to the sub types. This casting is generally done explicitly.

Question 10

Which is the exception handler for the exceptions of downcasting?

a) CastException
b) ClassCastingExeption
c) ClassCasting
d) ClassCastException

Question 10

Which is the exception handler for the exceptions of downcasting?

a) CastException
b) ClassCastingExeption
c) ClassCasting
d) ClassCastException

Explanation: The exception handler for the exceptions produced during the downcasting exception. This handler can be called during runtime to handle any exception thrown.

Question 11

Which way the downcasting is possible with respect to inheritance?

a) Upward the inheritance order
b) Downward the inheritance order
c) Either upward or downward the inheritance order
d) Order of inheritance doesn’t matter

Question 11

Which way the downcasting is possible with respect to inheritance?

a) Upward the inheritance order
b) Downward the inheritance order
c) Either upward or downward the inheritance order
d) Order of inheritance doesn't matter

Explanation: The downcasting is always downward the inheritance order. Since the base class object have to be casted into derived class type. This is a basic definition of downcasting.