- COMP90041-Programming-and-Software-Development
- Author: Chongzheng Zhao(https://github.com/ChongzhengZhao/)

1, Why instance variables should be declared PRIVATE?

- if not private, the class cannot control access and modification of instance variables, because other classes also can access and modify them.
- if not private, it is expensive to remove or change them, because making it public will require significant modification to other classes.

2, Class variable is what?

- A class variable is any variable declared with "static". It is a special type of class attribute.

3, Println mechanism for why it can print any any any objects

- "System" class has an instance variable called "out".
- the "out" class defines "println" method.
- the method is overloaded to support all primitive types, String and objects.
- for passing objects to the method, it will use object's "toString" method to represent the objects and send the representation to the output stream.

4, Static method VS Non-static method.

- Static method do not have "this" parameter
- Because it can be invoked without sending a message to any object
- it prevents many actions that non-static method can performe.
- prevent e.g. access, set instance variables of the class
- prevent e.g. call other non-static methods of the class

5, Constructor do what? Why usually overloaded?

- Constrctor is for initializing the instance variables to appropriate value.

- Make JAVA polymorphism: an ad hoc feature. It allows a single piece of code to work for many different types of objects.
- Default(no-argument constructor):An empty constructor is needed to create a new instance via reflection by your persistence framework.

6, Abstract class VS Concrete class
- Abstract Class:
- Abstract class permits to have abstract methods(method with header but no bodies).
- Used to specify an interface without implementing any methods.
- Not permit to create an instance of abstract class
- Allow extend from another abstract class
- Concrete Class:
- Concrete class not permits to have abstract methods.
- Concrete Class can extend abstract class when specify and override all abstract methods and even add other concrete methods.

7, Overloading VS Overriding
- Overloading:
- with same method name but different signatures in one class
- Overridding:
- derived class redefines an inherited method in the base class, with same signature.

8, Visibility of public VS protected.
- public members can be accessed from any methods in any classes
- protected member can only be accessed from the same package and any other classes that inherited from the member's class.
- CAN GIVE EXAMPLE

9, List all Visibility(Permission) modifiers. which one is prefered visibility for instance variables?
- Public, Protected, Private(preferred in

instance variables)

10, Single inheritance VS Multiple inheritance
- Single inheritance:
- each class can be derived from at most one class.
- JAVA onely support this because only extend one class.
- Multiple inheritance:
- each class can be derived from more than one class.
- JAVA can implement more than one interfaces, however it cannot be inherited for method function, only interface(it is abstract).

11, Why output excute the actual body instead of the type?(Type NAME = new ACTUALTYPE())
- An upcasting changes its type only, not change its actual body.
- the function will use the method in actual body.
- It is called LATE BINDING!

12, Privacy leak. Why leak although declare Private? How to prevent? Example?
- Privacy leak happened when method of another class get ahold of mutable objects stored in private instance variables.
- Prevent:
- Use an immutable object, such as String to hold data.
- Using copy constrctor to copy mutable object before storing or returning it.
- Simply do not any such methods to store or return such objects in instance variables.

13, What is polymorphism? why use? how use? example?
- Polymorphism is the feature that allows a single piece of code to work for many different types of objects
- JAVA use polymorphism in three ways: Ad hoc(overloading), Subtype(inheritance & overridding), Parametric(generic).

- SAY EACH FUNCTION OF THEM AND GIVE A PIECE OF CODE FOR EXAMPLE

14, Generic type - is what? when support in JAVA? improve what in Arraylist in former version of JAVA?

- generic types are types that have parameters
- for example: ArrayList<String> carry all the elements with type String
- it better specify the program intensions
- allow JAVA compiler to produce more specific error message when intension is violated.
-
- Generic type example - ArrayList
- Before JAVA 5, elements with any types can input to an arraylist
- each objects from arraylist need to be cast to an appropriate data type
- sometime it will cause runtime problem if choose wrong cast type
- But in JAVA 5, the type of arraylist can be specified
- no need to cast types like that because only one type will come out of an arraylist.

15, Wrapper Class is what? When useful? Example of a standard wrapper class.

- each primitive types have wrapper class which stores one primitive value
- each has a one-argument constrctor to create the object(boxing)
- each has a no-arugment getter to get the primitive value(unboxing)
- wrapper classes are immutable
-
- Used to convert any data type into an object
- Like: integer class is a wrapper class for the primitive type int which contains several methods to deal with int value like converting it to a string or convert from string. An object of Integer class can hold a single int value.

- Author: Chongzheng Zhao(https://github.com/ChongzhengZhao/)