API( Application programming interface)

Abstract class vs. interface

Abstract- only can extend 1 class, but implement many interfaces (method header w/out implementation, class w/ ab meth must be abstract, must be extended provide implementation.)

Can declared field,( but interface are auto public, static, final and all meths declare and define are public)

Common Supper Class-Object, method defined in super-ancestor- toString()

Collections methods - A Set implements only the methods inherited from Collection, but adds the additional restriction that duplicate elements are prohibited.

Class vs. Object

Object- created from class is an instance of that class,data & behavior, allocation in memory, physical exist, created as many time require

An object encapsulates data and behaviour.

\_ Data

\_ State

\_ Properties

In Java: \_elds, aka

member variables

\_ Behaviour

\_ Actions

\_ Activities

In Java: methods

Class- blueprint from which objects are created, no allocation, logical existence, defined once

Dynamic Binding vs. Polymorphism

Compiler don’t know what definition it will use later, current obj’s place inheritance chain not by the type naming the current object. Student myStudent = (Student) new Undergrad();

myStudent.toString(); will still call the Undergrad

toString() method.

Dynamic Binding is the compiler's way of realizing

Polymorphism

Empty –class (Has inherited methods and \_elds from parent

(in this case, Object) Has default constructor

try { clause for testing potential exception

code.

\_ catch { catching the exceptions, if they

happen

\_ throws { used in method headers to indicate

method might cause exception

\_ throw { used by a method to \throw" (cause)

an exception

\_ finally { code executed after the try-catch

clauses, regardless of whether exception

happened or not.

Ex. Rutime- ArrayIndexOutOfBoundsException

public class Sleeper {

public void sleep10Secs () {

try {

Thread . sleep (10000);

} catch ( InterruptedException ie ) {

System . out . println ( " Woke up early !" );

Polymorphism is at the object level (for the

programmer) describe a variable that may

refer to objects whose class is not known

at compile time, and which respond at run

time according to the actual class of the

object to which they refer.

Boolean (1 bit), char(2by), int(4by), float(4by), double(8by), short(2by), long(8by), byte(1by)

Overriding- pass the same name and parameter as parent, only to accessible methods, public final can’t be overridden, static is hidden. Useful for want inherit from parent but allow modifying when necessary.

Overloading- same name, diff parameter, useful when want to take in a char or String doing the same thing.

this a reference to the current object instance.

super a reference to the (parent) object that created this object instance. Use super to access methods that you are overriding in the

parent

public Accessible to all( class declared, subclass, same package, nested, all other classes)

private Only this class (nested class)

protected Only this class and its subclasses (the class, subclass, same package)

package-private No modifier. This class and others

in same package. (class, same package, subclass)

instanceOf test the A subclass can be assigned to a variable of the supertype class.(class, interface, too many polymorphism).

A subclass can be assigned to a variable of the

supertype class.

\_ If you want to check if it's one or the other,

you can use the instanceof keyword to check: Can be used to see if instances were created from the same class. <object> instanceOf <object>

interface- extreme abstract class, no headers, no implementations at all,class that implements must provide implementation for all methods, (comparable, collection, iterator, list

nested- A nested class is a member of its enclosing

class. Non-static nested classes are known as inner classes cannot directly access

Static nested- cannot directly access instance variables or methods in enclosing class. Need object reference.

An anonymous class is a local class need without a name.

static vs. non static

Static- 1 copy belong to class, change to every place used

Nonstatic- cant be access in class must be instance of object, 1 copy per instance.

1.if have constructor take param can’t use default constructor because the default doesn’t exist anymore unless both declared

2.Can make many mains inside classes for testing in isolation

3.example of collection size() returns the number of elements in this list.

4.recognize generic class API ( <T>)

5. using interface for many classes instead of extending superclass- superclass extend only once, can implement interfaces many times don’t care about the implemention as long as they implement all methods inside interface, super class use all the present implementation except little change.

5. 2 diff btwn abstract vs. interface 1. Only can extend a class once, can declare field,must extend abstract and implement all methods specify in interface.

6. multiple catch clauses order important because it executes the first catch clause over the ones after it. So if want to catch all execption than must include it in catch.

7. unchecked exceptions extend runtime and checked extend exception. Checked need to include method throws it, and try and catch. Runtime just throws exeception.

8. write method takes a List<String> and returns # of characters in all the Strings. (return sum of the lengths of the Strings in the List)

Public stat int ( List<String> strs){  
 int sum =0;

for(String s; strs){  
 sum += s.lengths();

}

return sum;

chang to generic public static (collection<string> strs)

public class Foo {

protected int val;

protected String name ;

public Foo () {

this (" BLAH ");

}

public Foo ( String name ) {

this ( name . length (), name );

}

private Foo (int val , String name ) {

this .val = val ;

this . name = name ;

}

public void printStuff (int val ) {

System .out . println (val );

System .out . println ( name );

}

public void printStuff ( String name ) {

System .out . println (val );

System .out . println ( name );

}

}

public class Bar extends Foo {

public Bar ( String name ) {

System .out . println ( name );

}

public void printStuff (int val ) {

super . printStuff (val );

System .out . println ( this .val );

}

public static void main ( String [] args ) {

Foo test = new Bar("MEH");

test . printStuff (37);

test . printStuff ("BOO ");

}

}

Output: MEH 37 BLAH 4 4 BOO

public class Baz {

private static int x;

private int y;

public Baz (int val ) {

x = val;

y = val \*2;

}

public void printStuff () {

System .out . println (x);

System .out . println (y);

}

public static void main ( String [] args ) {

Baz b1 = new Baz (2);

b1. printStuff ();

Baz b2 = new Baz (3);

b1. printStuff ();

b2. printStuff ();

}

}

Output: 2 4 3 4 3 6

public class WhileLoopExample {

public static void main ( String [] args ) {

int j = 100 , sum = 0;

while ( j > 0 ) {

sum += j --;

}

System .out . println ( "Sum is: " + sum

}

public class DoWhileLoopExample {

public static void main ( String [] args ) {

Scanner sc = new Scanner ( System .in );

int num = 0;

do {

System . out . print (" Enter a number [1 -100]: ");

num = sc. nextInt ();

} while ( num < 1 || num > 100 );

public class ForLoopExample {

public static void main ( String [] args ) {

for ( int i = 0; i < 10; i++ ) {

System . out . println ( "i = " + i );

public class ForLoopExample {

public static void main ( String [] args ) {

for ( String arg : args ) {

System . out . println (arg );

Quiz1

1. Best way to store pi = float
2. Modifier when want entire class to share only one copy= static
3. Protected variable visible to the class which it is declared and classes that extended it
4. Can change the modifier of a method in a subclass
5. Okay to allow more acess than parent
6. Cant make method less accessible than parent
7. Can acess private method in subclass

Quiz 2

1. Order of modifiers from least visible to most

1.private,no access modifier, protected, public

(only difference btwn no access and protected is that protected makes parent’s private members visible to nested class)

1. Public abstract final class- makes no sense to create abstract final class since abstract class must be extended and final can’t be extended.
2. Complier error “non-static variable X cannot be referenced from a static context”. Fix by create an instance of the object containing the non-static members you are trying to reference. Then access the member through the instance.
3. Public class Foo{

Protected int x;

}

Public Foo(int x){  
 this.x =x;

}

}

Public class Bar extends Foo{  
 public Bar(int x){  
 this.x = x;

}

}

1. Anonymous inner class – a class both defined and instantiated in a single expression w/ a method.