**Tarsem Monga #300137251**

**Chapter 1:**

**In short**

**Q1. What extensions are used for Java source and compiled files?**

**Ans.** .class for compiled java source code file.

.java for source file.

**Q2. Describe the three kinds of comments used in Java programs.**

**Ans.**

1. Single line comments
2. Multi-line comments
3. Javadoc comments

**Q3. What are the eight primitive types in Java?**

**Ans.** Primitives in java are byte, int, long, short, Boolean, float, char and double.

**Q4. What is the difference between the \* and \*= operators?**

**Ans.** \* means multiplication. For example: 5 \* 5.

\*= means the multiplication of the left value to right. For example x \*= y

It means x = x\*y.

**Q5. Explain the difference between the prefix and postfix increment operators.**

**Ans.**

**Postfix:** passes the current value of n to the function and then increments it.

**Prefix:** increments the current value of n and then passes it to the function.

**In theory**

**Q1. Let b have the value of 5 and c have the value of 8. What is the value**

**of a, b, and c after each line of the following program fragment:**

**a = b++ + c++;**

**a = b++ + ++c;**

**a = ++b + c++;**

**a = ++b + ++c;**

**Ans.** After line 1, a is 13, b is 6 and c is 9. Line 2, a is 16, b is 7 and c is 10. After line 3, a is 18 , b is 8 and c is 11. In the end, a is 21, b is 9 and c is 12 .

**Q2. What is the result of true && false || true?**

**Ans.** Result will be true, because of precedence rules, the expression implies as (true && false) || true.

**Q3. For the following, give an example in which the for loop on the left is**

**not equivalent to the while loop on the right:**

**init;**

**for( init; test; update ) while( test )**

**{ {**

**statements statements**

**update;**

**} }**

**Ans.**

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

system.out.println(“the value is ” + i);

}

init a = 100;

while (a>0){

system.out.println(a);

a/=2;

}

**Q4. For the following program, what are the possible outputs?**

**public class WhatIsX**

**{**

**public static void f( int x )**

**{ /\* body unknown \*/ }**

**public static void main( String [ ] args )**

**{**

**int x = 0;**

**f( x );**

**System.out.println( x );**

**}**

**}**

**Ans.**  the x will be 0, because after call to the method f of x. Output will be zero.

**Chapter 2:**

**Q1. List the major differences between reference types and primitive types.**

**Ans.** Reference list: it store the address of the object. Reference types are instantiable class as well as array. Primitive value store the value of a primitive variable. Primitives types are basic data like float, short. == has different meaning for both types.

**Q2. List five operations that can be applied to a reference type.**

**Ans.** assignment via=, comparison via=, !=, the operator, type conversion and instanceof.

**Q3. What are the differences between an array and ArrayList?**

**Ans.**An ArrayList has a capacity to store, array has its size associated with it. if you add any element to the ArrayList, it will automatically expand the size. we cannot change the length of array once it is created.

**Q4. Describe how exceptions work in Java.**

**public static void foo( )**

**{**

**try**

**{**

**return 0;**

**}**

**finally**

**{**

**return 1;**

**}**

**}**

**public static void bar( )**

**{**

**try**

**{**

**throw new NullPointerException( );**

**}**

**finally**

**{**

**throw new ArithmeticException( );**

**}**

**Ans.** Exceptions are handled by the blocks of the code. The try block and the catch block. the exception propagates back through the calling sequence until handled by the matching catch time.

**Q5. List the basic operations that can be performed on Strings.**

**Ans.** Equals, compareTo, =to copy, + and += to perform concatenation, substring, length and charAt.

**in Theory**

**Q1. If x and y have the values of 5 and 7, respectively, what is output by**

**the following?**

**System.out.println( x + ' ' + y );**

**System.out.println( x + " " + y );**

**Ans.** The first output is 44 and second output is 5 7.

**Q2. The finally block introduces complications in the Java language specification. Write a program to determine what value is returned by foo and what exception is thrown by bar in Figure 2.21.**

**Ans.** public class Test {

public String s;

public String foo() {

try {

s = "dev";

return s;

} finally {

s = "override variable s";

System.out.println("Entry in finally Block");

return s;

}

}

public static void main(String[] xyz) {

Test obj = new Test();

System.out.println(obj.foo());

System.out.println(obj.s);

}

}

**Chapter 3:**

**Q1. What is information hiding? What is encapsulation? How does Java support these concepts?**

**Ans.** Encapsulation is the grouping of data and the operations that apply to them to form an aggregate while hiding the implementation of the aggregate. information hiding makes implementation details, including components of an object. encapsulation and information hiding are achieved in java through the use of the class.

**Q2. Explain the public and private sections of the class.**

**Ans.** private sections are not visible outside of the class, but in public section are visible to non-class routines and can be accessed through dot operator.

**Q3. Describe the role of the constructor.**

**Ans.** the constructor is called when an object is created by a call to new.

**Q4. If a class provides no constructor, what is the result?**

**Ans.** the default constructor is a member by member application of a default constructor. in other words,

**Q5. Explain the uses of this in Java.**

**Ans.** this is a reference to the current object. this reference to the current object can be used to compare it with other objects or to pass the current object to the some other unit.

**in theory**

**Q1. A class provides a single private constructor. Why would this be useful?**

**Ans.** single private constructor for class A can disallow any other object to create an instance A.

**Q2.Suppose that the main method in Figure 3.3 was part of the IntCell class.**

**a. Would the program still work?**

**Ans.** Yes, main works anywhere.

**b. Could the commented-out line in main be uncommented without generating an error?**

**Ans.** yes, if main was part of class Int-Cell then StoredValue would no longer be considered private to main.

**Q3. Is the following import directive, which attempts to import virtually the entire Java library, legal?**

**import java.\*.\*;**

**Ans.) No**

**Chapter 4:**

**4.1 What members of an inherited class can be used in the derived class? What members become public for users of the derived class?**

**Ans.** Assuming public inheritance, only public and protected members of the base class are visible in the derived class unless the derived class is in the same package of the base class, in which case package-friendly members are also visible. Among base class members, only public members of the base class are visible to users of the derived class.

**4.2 What is composition?**

**Ans.** Private inheritance is used to indicate a HAS-A relationship while avoiding the overhead of a layer of function calls. Composition is a better alternative for this. In composition, a class is composed of objects of other classes.

**4.3 Explain polymorphism. Explain dynamic dispatch. When is dynamic dispatch not used?**

**Ans.** Polymorphism is the ability of a reference type to reference objects of several different types. When operations are applied to the polymorphic type, the operation that is appropriate to the actual referenced object is automatically selected

**4.4 What is autoboxing and unboxing?**

**Ans.** these are the new features of java 1.5, by this we can perform automatic conversions between primitive types and their object parts.

**4.5 What is a final method?**

**Ans.** A final method is a method that cannot be redefined in a derived class.

**in Theory**

**Q1. Answer each part TRUE or FALSE:**

**a. All methods in an abstract class must be abstract.**

**False**

**b. An abstract class may provide constructors.**

**true**

**c. An abstract class can declare instance data.**

**False**

**d. An abstract class can extend another abstract class.**

**True**

**e. An abstract class can extend a non-abstract class.**

**True**

**f. An interface is an abstract class.**

**false**

**g. An interface can declare instance data.**

**False**

**h. Any method in an interface must be public.**

**true**

**i. All methods in an interface must be abstract.**

**False**

**j. An interface can have no methods at all.**

**True**

**k. An interface can extend another interface.**

**false**

**l. An interface can declare constructors.**

**true**

**m. A class may extend more than one class.**

**false**

**n. A class may implement more than one interface.**

**false**

**o. A class may extend one class and implement one interface.**

**false**

**p. An interface may implement some of its methods.**

**true**

**q. Methods in an interface may provide a throws list.**

**True**

**r. All methods in an interface must have a void return type.**

**true**

**s. Throwable is an interface.**

**false**

**t. Object is an abstract class.**

**False**

**u. Comparable is an interface.**

**true**

**v. Comparator is an example of an interface that is used for function objects.**

**true**

**Q2. Carefully examine the online documentation for the Scanner constructors. Which of the following are acceptable parameters for a Scanner: File, FileInputStream, FileReader.**

**Ans.** For scanner acceptable parameter is FileInputStream.

**Q3. A local class can access local variables that are declared in that method (prior to the class). Show that if this is allowed, it is possible for an instance of the local class to access the value of the local variable, even after the method has terminated. (For this reason, the compiler will insist that these variables are marked final.)**

**Ans.** Consider the case in which a method f o o ( ) contains a local class and returns an object of type L o c a l C l a s s . In this case, if the local class accesses x, then it continues to be available after foo terminates.

Typetest foo ()

{

final int x = 1;

class LocalClass() implements Typetest

{

public int foo()

{

return x; }

}

return new localClass();

}

**Q4. This exercise explores how Java performs dynamic dispatch, and also**

**why trivial final methods may not be inlined at compile time. Place**

**each of the classes in Figure 4.49 in its own file.**

**a. Compile Class2 and run the program. What is the output?**

**b. What is the exact signature (including return type) of the getX**

**method that is deduced at compile time at line 14?**

**c. Change the getX routine at line 5 to return an int; remove the ""**

**from the body at line 6, and recompile Class2. What is the output?**

**d. What is the exact signature (including return type) of the getX**

**method that is now deduced at compile time at line 14?**

**e. Change Class1 back to its original, but recompile Class1 only.**

**What is the result of running the program?**

**f. What would the result have been had the compiler been allowed**

**to perform inline optimization?**

**Ans.** a.) The output is 512.

b.) The signature is String getX()

C.) After some changes the output will be 17 and signature will be int getX().

d.) After class 1 is changed we will get error which will be class 2 is expecting a method getX returning an

integer.

e.)If code inlining was allowed then the output will be 17.

**Q5. In each of the following code fragments, find any errors and any**

**unnecessary casts.**

**a. Base [ ] arr = new Base [ 2 ];**

**arr[ 0 ] = arr[ 1 ] = new Derived( );**

**Derived x = (Derived) arr[ 0 ];**

**Derived y = ( (Derived[])arr )[ 0 ];**

**b. Derived [ ] arr = new Derived [ 2 ];**

**arr[ 0 ] = arr[ 1 ] = new Derived( );**

**Base x = arr[ 0 ];**

**Base y = ( (Base[])arr )[ 0 ];**

**c. Base [ ] arr = new Derived [ 2 ];**

**arr[ 0 ] = arr[ 1 ] = new Derived( );**

**Derived x = (Derived) arr[ 0 ];**

**Derived y = ( (Derived[])arr )[ 0 ];**

**d. Base [ ] arr = new Derived [ 2 ];**

**arr[ 0 ] = arr[ 1 ] = new Base( );**

Ans. a.) The last line throws a ClassCaseExcption as the element in array is not specificallty casted.

b.)The last line contains un necessary casts.

c.) No errors

d.) No errors.