=

****

**INTERNATIONAL CENTER FOR APPLIED SCIENCES**

(A constituent college of Manipal University, Manipal)

Manipal Karnataka 576 104

**II SEMESTER B.S.(CS) II SESSIONAL TEST**

**SUBJECT: JAVA PROGRAMMING (CS 124)**

**Max Marks: 20 Date: 25/03/2017 Duration: 1 hour**

**Instructions:**

* **Answer any Four full QUESTIONS**
* **Leave the first page of the Answer Book Blank**

1. Write any 5 differences between interfaces and abstract classes. 5M

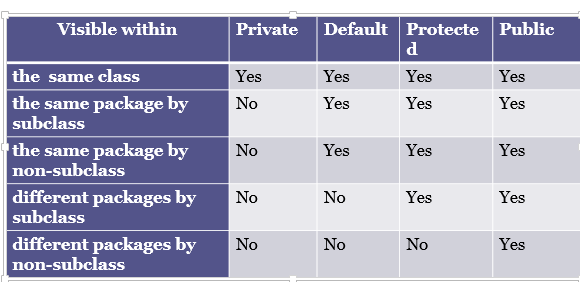
|  |  |  |
| --- | --- | --- |
|  | Abstract classes | Interfaces |
| 1 | abstract  class  can  have  both  abstract and concrete methods. | interface can  have only abstract methods |
| 2 | Can have instance variables | Can have only static final variables ( constants) |
| 3 | Can have constructors | Cannot have constructors |
| 4 | Should be extended by another class | Should be implemented by a class |
| 5 | **Example:** public abstract class Shape  {  public abstract void draw(); }  abstract keyword compulsory here | **Example:**  public interface Drawable{ void draw(); }  abstract keyword is not compulsory here |

**2A) Mention the two benefits of packages. 2M**

The benefits are –

* Classes in packages can be easily reused
* Two different packages can contain classes with same name
* Provide a way to ‘hide’ classes thus preventing other programs or packages from accessing classes that are meant for internal use only

**2B) Write a table showing the class members visibility with respect to packages. 3M**



1. **What are Exceptions? What are checked and unchecked exception? Give two examples for each.**

An exception is an abnormal condition that arises in a code sequence at run-time (run-time error) 1M

**1. Checked exceptions 2M**

* All exceptions **other than Runtime Exceptions** are known as Checked exceptions as the compiler checks them during compilation to see whether the programmer has handled them or not.
* If these exceptions are not handled/declared in the program, it will give compilation error.

Examples :-

ClassNotFoundException, IllegalAccessException, NoSuchFieldException

EOFException

**2. Unchecked Exceptions 2M**

* Runtime Exceptions are also known as Unchecked Exceptions as the compiler do not check whether the programmer has handled them or not but it’s the duty of the programmer to handle these exceptions and provide a safe exit.
* These exceptions need not be included in any method’s throws list because compiler does not check to see if a method handles or throws these exceptions.

Examples :-

ArithmeticException, ArrayIndexOutOfBoundsException, NullPointerException

4.a) What is the difference between process-based multitasking and thread-based multitasking?

Ans:

* In a *thread-based* multitasking environment, the thread is the smallest unit of dis-patchable code
* Multitasking threads require less overhead than multitasking processes
* Processes are heavyweight tasks that require their own separate address spaces
* Inter-process communication is expensive and limited
* Context switching from one process to another is also costly 2M

b) Write a program to demonstrate multiple threads. Each thread should display numbers from 1 to 10 in an interleaved fashion. Use proper exception handling mechanism.

Ans:  
class NewThread implements Runnable {

String name; // name of thread

Thread t;

NewThread(String threadname) {

name = threadname;

t = new Thread(this, name);

System.out.println("New thread: " + t);

t.start(); // Start the thread

} (1M)

// This is the entry point for thread.

public void run() {

try {

for(int i = 1; i <=10; i++) {

System.out.println(name + ": " + i);

Thread.sleep(1000);

}

}

catch (InterruptedException e) {

System.out.println(name + "Interrupted");

}

System.out.println(name + " exiting.");

}

} (1M)

class MultiThreadDemo {

public static void main(String args[]) {

new NewThread("One"); // start threads

new NewThread("Two");

new NewThread("Three");

try {

// wait for other threads to end

Thread.sleep(10000);

}

catch (InterruptedException e) {

System.out.println("Main thread Interrupted");

}

System.out.println("Main thread exiting."); } (1M)

}

5) a) What is the need for synchronization? With an example program explain synchronized

method. 3M

Ans:

When two or more threads need access to a shared resource, they need some way to ensure that the resource will be used by only one thread at a time. The process by which this is achieved is called synchronization (1M)

**Program:**

lass M implements Runnable{

int x;

multi m;

Thread t;

public M(multi m1, int x1) {

x=x1;

m=m1;

t=new Thread (this);

t.start();

}

public void run() {

m.mul(x);

}

}

class multi{

synchronized void mul(int x) {

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

System.out.println(x +"\*"+(i+1)+"="+(x\*(i+1)));

Thread.sleep(1000);

}

} catch(InterruptedException e) {

System.out.println("Interrupted");

}

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

}

}

class multiply {

public static void main(String args[]) {

multi m= new multi();

M ob1 = new M(m,5);

M ob2 = new M(m,7);

}

} (2M)

b) Differentiate between join() and sleep() methods. 2 M

**(i) sleep()**

**Syntax:** void sleep(int )

* The argument to sleep( ) specifies the delay period in milliseconds
* Is used to suspend the thread for some time

**(ii) join()**

**Syntax:** void join() throws InterruptedException

* This method waits until the thread on which it is called terminates

1. a) What are Bounded and Unbounded wildcards? Explain with the help of an example

program. 3M

Ans:

Any Type can be bounded either upper or lower of the class hierarchy in Generics by using bounded wildcards.

In short <? extends T> and <? super T> represent bounded wildcards while <?> represent an unbounded wildcard in generics . (1M)

class GenericStats<T extends Number> {

T[] nums;

GenericStats(T[] o) {

nums = o;

}

double average() {

double sum = 0.0;

for(int i=0; i < nums.length; i++){

sum += nums[i].doubleValue();

} return sum / nums.length;

}

boolean sameAvg1(GenericStats<?> ob) {

if(average() == ob.average())

return true;

return false;

}

boolean sameAvg2(GenericStats<? extends Integer > ob) {

if(average() == ob.average())

return true;

return false;

}

}2M

b) With examples mention any 2 restrictions that we need to keep in mind when using

generics. 2M

Ans: (Any 2)

(i)Static variables and methods cannot use the type parameters declared by their generic class (but you can have static generic methods).

Example:

class C<T> {

static T x; // illegal

static T getX() { return x; } // illegal}

(ii)You can't create an instance of a type parameter.

Example:

class C<T>

{

T x;

C()

{ x= new T(); // illegal

}

}

(iii) You can't create an array whose element type is the type parameter nor an array of type-specific generic references.

Example:

class C<T> {

T[] x ;

x = new T[10]; // illegal

C<String>[] data= new C<String>[10]; // illegal

C<?>[] data = new C<?>[10]; // legal

}

\*\*\*\*\*GOODLUCK\*\*\*\*\*