**Core Java: Part 3**

**1. Determine the output**

```
PSVM()

{

try{

int a = 5;

int b = 0;

int c = a/b;

SOP(“World”);

}

Catch(exception e)

{

SOP(“hello”);

}}
```

**ANS:hello**

**2. What is Proper order of access modifier**

a) private default protected public

b) default private protected public

c) public private default protected

d) public private protected default

ANS:b) default private protected public

**3 .The code snippet below is an example of which of the following?**

**Long myLong = 21l;**

A Autoboxing

B Autounboxing

C Autocasting

D Autoinstancing

**ANS:A Autoboxing**

**4.**

**1. interface TestA { String toString(); }**

**2. public class Test {**

**3. public static void main(String[] args) {**

**4. System.out.println(new TestA() {**

**5. public String toString() { return "test"; }**

**6. });**

**7. }**

**8. }**

**What is the result?**

A. test

B. null

C. An exception is thrown at runtime.

D. Compilation fails because of an error in line 1.

E. Compilation fails because of an error in line 4.

F. Compilation fails because of an error in line 5.

ANS:A. test

**5.Determine the output**

**int a = 9;**

**int b = 14;**

**while(a<b) {**

**System.out.println("In the loop");**

**a+=2;**

**b-=2;}**

a)In the loop

In the loop

b)In the loop

c)none of the above

ANS:a)In the loop

In the loop

**6.What is the output of this program?**

**Import java.util.***

**Public static void main(String args[]){**

**TreeMap obj = new Treemap();**

**Obj.put(“A”, newInteger(1));**

**Obj.put(“b”, newInteger(2));**

**Obj.put(“c”, newInteger(3));**

**SOP(obj.entrySet());**

**}**

a)[A=1, b=2, c=3]

b)[A=1, B=2, C=3]

c)will not execute

ANS:b)[A=1, B=2, C=3]

**7.Which implementation of set would you choose if you want the iterator of set would give you object in the order it were inserted?**

a)LinkedHashSet

b)TreeSet

c)HashSet

ANS:a)LinkedHashSet

**8. How can you retrieve information from a ResultSet?**

(a) By invoking the method get(..., String type) on the ResultSet, where type is the database type

(b) By invoking the method get(..., Type type) on the ResultSet, where Type is an object which represents a database

type

(c) By invoking the method getValue(...), and cast the result to the desired Java type.

(d) By invoking the special getter methods on the ResultSet: getString(...), getBoolean (...),

getClob(...),...

ANS:(d) By invoking the special getter methods on the ResultSet: getString(...), getBoolean (...),getClob(...),...

**9.Determine the output**

```
import java.util.*;

class TestHashMaps{

public static void main(String args[]) {

HashMap<Integer,String> hm= new HashMap<Integer,String> ();

hm.put(100, "John");

hm.put(101, "Paul");

hm.put(102, "George");

hm.put(103, "Ringo");

for (Map.Entrym: hm.entrySet()) {System.out.println(m.getKey() + " " + m.getValue());

}

}

}
```

a) 100 John

101 Paul

102 George

103 Ringo

b)103 Ringo

102 George

101 Paul

100 John

c)none of the above

ANS:a) 100 John

101 Paul

102 George

103 Ringo

**10.Determine the output**

```
import java.util.Map;

import java.util.TreeMap;

public class TestTreeMap {

public static void main(String args[]) {

TreeMap< Integer, String > hm= new TreeMap< Integer, String > ();

hm.put(100, "John");

hm.put(102, "Paul");

hm.put(101, "George");
```

```
hm.put(103, "Ringo");

for (Map.Entry m: hm.entrySet()) {

System.out.println(m.getKey() + " " + m.getValue());

}

}

}
```

a)100 John

101 George

102 Paul

103 Ringo

b) 100 John

101 Paul

102 George

103 Ringo

c)103 Ringo

102 George

101 Paul

100 John

d)none of the above

ANS:a)100 John

101 George

102 Paul

103 Ringo

**11.What is the result?**

**5. import java.util.*;**

**6. public class SortOf {**

**7. public static void main(String[] args) {**

**8. ArrayList<Integer> a = new ArrayList<Integer>();**

**9. a.add(1); a.add(5); a.add(3);11. Collections.sort(a);**

**12. a.add(2);**

**13. Collections.reverse(a);**

**14. System.out.println(a);**

**15. }**

**16. }**

A. [1, 2, 3, 5]

B. [2, 1, 3, 5]

C. [2, 5, 3, 1]

D. [5, 3, 2, 1]

E. [1, 3, 5, 2]

F. Compilation fails.

G. An exception is thrown at runtime.

ANS:C. [2, 5, 3, 1]

**12. class BabyRaccoon extends Mammal { }**

**Which four statements are true? (Choose four.)**

A. Raccoon is-a Mammal.

B. Raccoon has-a Mammal.

C. BabyRaccoon is-a Mammal.

D. BabyRaccoon is-a Raccoon.

E. BabyRaccoon has-a Mammal.

F. BabyRaccoon is-a BabyRaccoon.

ANS:A,B,C,F

**13. Which Man class properly represents the relationship "Man has a best friend who is**

**a Dog"?**

A. class Man extends Dog { }

B. class Man implements Dog { }

C. class Man { private BestFriend dog; }

D. class Man { private Dog bestFriend; }

E. class Man { private Dog<bestFriend>; }

F. class Man { private BestFriend<dog>; }

ANS:D. class Man { private Dog bestFriend; }

**14.What is the result**

```
11. class Alpha {
12. public void foo() { System.out.print("Afoo "); }
13. }
14. public class Beta extends Alpha {
15. public void foo() { System.out.print("Bfoo "); }
16. public static void main(String[] args) {
17. Alpha a = new Beta();
18. Beta b = (Beta)a;
19. a.foo();
20. b.foo();
21. }
22. }
```

A. Afoo Afoo

B. Afoo Bfoo

C. Bfoo Afoo

D. Bfoo Bfoo

E. Compilation fails.

F. An exception is thrown at runtime.

ANS:B. Afoo Bfoo

**15. Which code fragment, inserted at line 23, allows the code to compile?**

**5. import java.util.Date;**

**6. import java.text.DateFormat;**

**21. DateFormat df;**

**22. Date date = new Date();**

**23. // insert code here**

**24. String s = df.format(date);**

A. df = new DateFormat();

B. df = Date.getFormat();

C. df = date.getFormat();

D. df = DateFormat.getFormat();

E. df = DateFormat.getInstance();

ANS:E. df = DateFormat.getInstance();

**16. What is the result?**

**1. public class Base {**

**2. public static final String FOO = "foo";**

**3. public static void main(String[] args) {**

**4. Base b = new Base();**

**5. Sub s = new Sub();**

**6. System.out.print(Base.FOO);**

**7. System.out.print(Sub.FOO);**

**8. System.out.print(b.FOO);**

**9. System.out.print(s.FOO);**

**10. System.out.print(((Base)s).FOO);**

**11. } }**

**12. class Sub extends Base {public static final String FOO="bar";}**

A. foofoofoofoofoo

B. foobarfoobarbar

C. foobarfoofoofoo

D. foobarfoobarfoo

E. barbarbarbarbar

F. foofoofoobarbar

G. foofoofoobarfoo

ANS:D. Foobarfoobarfoo

**17. A company has a business application that provides its users with many different**

**reports:**

**receivables reports, payables reports, revenue projects, and so on. The company has just**

**purchased some new, state-of-the-art, wireless printers, and a programmer has been**

**assigned the**

**task of enhancing all of the reports to use not only the company's old printers, but the new**

**wireless printers as well. When the programmer starts looking into the application, theprogrammer**

**discovers that because of the design of the application, it is necessary to make changes to**

**each**

**report to support the new printers. Which two design concepts most likely explain this**

**situation?**

**(Choose two.)**

A. Inheritance

B. Low cohesion

C. Tight coupling

D. High cohesion

E. Loose coupling

F. Object immutability

ANS:B,C

**18.A team of programmers is reviewing a proposed API for a new utility class. After some**

**discussion,**

**they realize that they can reduce the number of methods in the API without losing any**

**functionality. If they implement the new design, which two OO principles will they be**

**promoting?**

A. Looser coupling

B. Tighter coupling

C. Lower cohesion

D. Higher cohesion

E. Weaker encapsulation

F. Stronger encapsulation

ANS:A,D

**19. A team of programmers is involved in reviewing a proposed design for a new utility**

**class. After**

**some discussion, they realize that the current design allows other classes to access methods**

**in**

**the utility class that should be accessible only to methods within the utility class itself. What**

**design**

**issue has the team discovered?**

A. Tight coupling

B. Low cohesion

C. High cohesion

D. Loose coupling

E. Weak encapsulation

F. Strong encapsulation

ANS:E. Weak encapsulation

**20. A programmer has an algorithm that requires a java.util.List that provides an efficient**

**implementation of add(0, object), but does NOT need to support quick random access. What**

**supports these requirements?**

A. java.util.Queue

B. java.util.ArrayList

C. java.util.LinearList

D. java.util.LinkedList

**ANS:D. java.util.LinkedList**

**21. What is the output of this program?**

```
import java.util.*;

class Collection_Algos {

public static void main(String args[])

{

LinkedList list = new LinkedList();

list.add(new Integer(2));

list.add(new Integer(8));

list.add(new Integer(5));
```

**list.add(new Integer(1));**

**Iterator i = list.iterator();**

**Collections.reverse(list);**

**Collections.shuffle(list);**

**while(i.hasNext())**

**System.out.print(i.next() + " ");**

**}**

**}**

a) 2 8 5 1

b) 1 5 8 2

c) 1 2 5 8

d) Any random order

ANS:d) Any random order

**22.Which of these methods are used to read in from**

**file?**

a) get()

b) read()

c) scan()

d) readFileInput()

ANS:b) read()

**23.What is the ouput of the below code?**

```
interface A{}

class C{}

class D extends C{}

public class Test extends D{

public static void main(String[] args) {

Test t = new Test();

if(t instanceof A){

System.out.println("instance of A");

}else if(t instanceof C)

{

System.out.println("instance of C");

}

else if(t instanceof D)

{

System.out.println("instance of D");

}

else{

System.out.println("Hello World");

}}
```

**}**

**}**

A) instance of A

instance of D

B) instance of C

instance of D

C) instance of C

D) instance of C

E) Compilation Fails

ANS:C) instance of C

D) instance of C

**24.class Parent{**

**void method(){**

**System.out.println("Parent");**

**}**

**}**

**class Child extends Parent{**

**void method(){**

**System.out.println("Child");**

**}**

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

**Parent p = new Parent();**

**Child c = (Child)p;**

**c.method();**

**}**

**}**

A) Child

B) Parent

C) Compilation fails

D) ClassCastException thrown at runtime

ANS:D) ClassCastException thrown at runtime

**25.Determine the output**

**class Animal**

**{**

**String name = "animal";**

**String makeNoise() { return "generic noise"; }**

**}**

**class Dog extends Animal**

**{**

**String name = "dog";**

**String makeNoise() { return "bark"; }**

```
}

public class Test

{

public static void main(String[] args)

{

Animal an = new Dog();System.out.println(an.name+" "+an.makeNoise());

}

}
```

A) animal generic noise

B) animal bark

C) dog bark

D) dog generic noise

ANS:B) animal bark