OOP'S EXAM

==========

1. Functional interfaces can be annotated as

- ◯ @FunctionalInterface
- ◯ @functionalinterface
- ◯ @Interface
- ◯ @Function

==========

2. Which of the following statements regarding abstract classes are true?

- ◯ An abstract class can be used as a data type.
- ◯ All of the above
- ◯ A subclass of a non-abstract superclass can be abstract.
- ◯ A subclass can override a normal method in a superclass to declare it abstract.
- ◯ An abstract class can be extended

==========

3. class A

```
{
   public void method1()
   {
      System.out.print("Class A method1");
   }
}
```

```
class B extends A
{
   public void method2()
   {
```

```
        System.out.print("Class B method2");
    }
}
----------------------------------------
class C extends B
{
    public void method2()
    {
        System.out.print("Class C method2");
    }
    public void method3()
    {
        System.out.print("Class C method3");
    }
}
-------------------------------------------------------
public class ClassA
{
    public static void main(String[] args)
    {
        A a = new A();
        C c = new C();
        c.method2();
        a = c;
        a.method3();
    }
}
```

- ○ Compilation Error
- ○ Class C method2 Class C method3
- ○ Runtime exception

- None of the these
- Class B method2 Class C method3

==========

4. public class ClassA

```
{
   static String value="abc";
   public static void changeValue(String a)
   {
      String data=a.concat(value);
      data= value;
   }
   public static void main(String[] args)
   {
      value = "Java";
      changeValue(value);
      System.out.println(value);
   }
}
```

- abcJava
- Java
- Compile time error
- abc
- None of the above

==========

5. What is the output for the below code ?

```
package com;
class Animal
{
```

```
    public void printName()
   {
       System.out.println("Animal");
   }
}
```

```
package exam;
import com.Animal;
public class Dog extends Animal
{
     public void printName()
    {
       System.out.println("Dog");
    }
}
```

```
package exam;
import com.Animal;
public class Test
{
     public static void main(String[] args)
    {
        Animal a = new Dog();
        a.printName();
    }
}
```

- ○ Compile time error
- ○ None of the above
- ○ Animal
- ○ Animal Dog

- ◯ Dog

==========

6. public class ClassA

```
{
   int i;
   boolean see;
   static int u;
   public void printValue()
   {
      System.out.print(i);
      System.out.print(see);
      System.out.print(u);
   }
}
```

```
public class ClassB
{
   public static void main(String[] args)
   {
      new ClassA().printValue();
   }
}
```

- ◯ 0true0
- ◯ None of the above
- ◯ 0false0
- ◯ Compile error - static variable must be initialized before use
- ◯ 000

==========

7. public class ClassA

```
{
```

```
    public static void main(String[] args)
    {
        byte b = 6;
        b+=8;
        System.out.println(b);
        b = b+7;
        System.out.println(b);
    }
}
```

- None of the above
- 21 14
- 14 13
- 14 21
- Compile time Error

==========

8. Suppose A is an abstract class, B is a normal subclass of A, and both A and B have a default constructor. Which of the following is correct?

1) A a = new A();

2) A a = new B();

3) B b = new A();

4) B b = new B();

- None of the above
- 2 and 3
- 1 and 3
- 1 and 2
- 3 and 4
- 2 and 4

==========

9. What is the output of the below code?

```
public class ClassA
{
   static{System.out.print("Java ");} //Static Block
   { System.out.print("is ");} // Instance Block (executes whenever we are creating an Object)
    public ClassA()
   {
      System.out.print("Awesome ");
   }
   public static void main(String[] args)
   {
      ClassA a = new ClassA();
   }
}
```

- Java Awesome is
- Awesome Java is
- Awesome
- Compile time error
- Java is Awesome

==========

10. Method overriding is combination of inheritance and polymorphism?

- True
- False

==========

11. What is the output?

```
public interface InfA
{
    protected String getName();
}
```

```
public class Test implements InfA
{
    public String getName()
  {
        return "Java ia awesome";
    }
    public static void main (String[] args)
  {
        Test t = new Test();
        System.out.println(t.getName().length());
    }
}
```

- test-name
- None of the above
- Compile time error
- 14

==========

12. Given the following piece of code:

```
public class School
{
     public abstract double numberOfStudent();
}
```

which of the following statements is true?

- You must add a return statement in method numberOfStudent().
- The keywords public and abstract cannot be used together
- All the above
- Class School must be defined abstract.
- The method numberOfStudent() in class School must have a body

==========

```
13.       class Super
{
      int value = 0;
      Super()
      {
         addValue();
      }
      void addValue()
      {
         value += 10;
      }
      int getValue()
      {
         return value;
      }
}
class Sub extends Super
{
   Sub()
   {
      addValue();
   }
   void addValue()
   {
      value +=  20;
   }
}
public class ClassA
{
```

```
    public static void main(String[] args)
    {
        Super b = new Sub();
        System.out.println(b.getValue());
    }
}
```

- 30
- 10
- 20
- 50
- 40

==========

14.

What is the output for the below code ?

```
public class A
{
    int i = 10;
    public void printValue()
    {
        System.out.println("Value-A");
    }
}
```

```
public class B extends A
{
    int i = 12;
    public void printValue()
    {
```

```
    System.out.print("Value-B");
  }
}
```

```
public class Test
{
 public static void main(String args[])
{
   A a = new B();
   a.printValue();
   System.out.println(a.i);
 }
}
```

- ○ Compile time error
- ○ Value-B 11
- ○ Value-B 10
- ○ Value-A 11
- ○ Value-A 10

==========

15.

```
class ClassA
{
  public void method()
  {
    System.out.println("Hi i am ClassA");
  }
}
```

```
public class ClassB extends ClassA
{
```

```
    protected void method()
    {
        System.out.println("Hi i am ClassB");
    }
    public static void main(String args[])
    {
        ClassB child = new ClassB();
        child.method();
    }
}
```

- None of the above
- Hi i am ClassB
- Compile time error
- Hi i am ClassA

==========

16. An interface with no fields or methods is known as a _____.

- Abstract Interface
- Functional Interface
- None of the above
- Runnable Interface

==========

17. Analyze the following code:

```
public abstract class Test implements Runnable
{
    public void doSomething() { };
}
```

- None of the options

- ○ The program will not compile because it does not implement the run() method.
- ○ The program will not compile because it does not contain abstract methods.
- ○ The program compiles fine

==========

18.

```
public class ClassA
{
   public static void main(String[] args){
      Mammal h = new Horse();
      Cattle c = new Horse();
      c.eat(h);
}
}
class Mammal{
   void eat(Mammal m){
       System.out.println("Mammal eats food");
   }
}
class Cattle extends Mammal{
   void eat(Cattle c){
       System.out.println("Cattle eats hay");
   }
}
class Horse extends Cattle{
   void eat(Horse h){
       System.out.println("Horse eats hay");
   }
}
```

- ○ prints "Cattle eats hay"
- ○ None of these

- prints "Mammal eats food"
- Exception
- prints "Horse eats hay"

==========

19. Which of the following class definitions defines a legal abstract class?

- public class abstract A { abstract void unfinished(); }
- class A { abstract void unfinished() { } }
- abstract class A { abstract void unfinished(); }
- None of the above
- class A { abstract void unfinished(); }

==========

20.
```
abstract class C1
{
   public C1(){
      System.out.print(1);
   }
}
class C2 extends C1{
   public C2(){
      System.out.print(2);
   }
}
class C3 extends C2{
   public C3(){
      System.out.println(3);
   }
}
public class Test{
```

```
    public static void main(String[] a){
        new C3();
    }
}
```

- 23
- 321
- Compile time error
- 123
- 12