|  |  |
| --- | --- |
| Course- BTech | Type- Core |
| Course Code- **CSET** | Course Name- **Object Oriented Programming Using Java** |
| Year- First | Semester- Even Batch- BTech 2nd Semester |

**Tutorial-6**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tutorial No.** | **Name** | **CO1** | **CO2** | **CO3** |
| **1** | **Basics** |  | **--** | **--** |

**Objective:** The main objective of this tutorial is to learn about the constructor overloading and functions overloading and overriding.

Q1. What is output of below code?

class Cn{  
 Cn() {  
 System.*out*.println("Cn constructor");  
 }  
 int getAge() {  
 return 19;  
 }  
}  
  
class Cm extends Cn{  
 Cm() {  
 System.*out*.println("Cm constructor");  
 }  
 int getAge() {  
 return super.getAge() +1;  
 }  
}  
  
class C1 extends Cm{  
 C1() {  
 System.*out*.println("C1 constructor");  
 }  
}  
  
class C0 extends C1{  
 C0() {  
 System.*out*.println("C0 constructor");  
 }  
}  
  
public class test {  
 public static void main(String[] args) {  
 C0 obj = new C0();  
 System.*out*.println(obj.getAge());  
 }  
}

Output –

Cn constructor

Cm constructor

C1 constructor

C0 constructor

20

Q2. What will be output of below code?

class Parent{  
 int par\_val;  
 Parent(int val) {  
 par\_val = val;  
 }  
  
 protected void display() {  
 System.*out*.println(par\_val);  
 }  
}  
  
public class test extends Parent {  
 int val;  
 test() {  
 this(10);  
 }  
 test(int init) {  
 super(init+2);  
 this.val = init;  
 }  
 protected void display() {  
 System.*out*.println(val + " " + par\_val);  
 }  
   
 public static void main(String[] args) {  
 test obj = new test(10);  
 obj.display();  
 Parent obj2 = new Parent(11);  
 obj2.display();  
 }  
}

Output –

10 12

11

Q3 What will be output of below code?

public class test {  
 int val;  
 test() {  
 this(10);  
 }  
 test(int init) {  
 if(init > 0)  
 this.val = init;  
 else  
 this.val = 0;  
 }  
 protected void display() {  
 System.*out*.println(val);  
 }  
   
 public static void main(String[] args) {  
 test obj = new test(13);  
 obj.display();  
 test obj1 = new test();  
 obj1.display();  
 test obj3 = new test(-5);  
 obj3.display();  
 }  
}

Output –

13

10

0

Q4 What will be effect of below code?

public class test {  
 public static int sumGP(int a, int r, int n, int term) {  
 if(n==1)  
 return term;  
 else {  
 return term + *sumGP*(a,r,n-1, term\*r);  
 }  
 }  
   
 public static void main(String[] args) {  
 System.*out*.println(*sumGP*(2,3,4,2));  
 }  
}

Output –

80

Q5 What is output of below code?

class Complex{  
 int x, y;  
 Complex(int a, int b) {  
 x = a;  
 y = b;  
 }  
  
 public void display() {  
 System.*out*.println("Real: " + x + ", Imaginary: " + y) ;  
 }  
}  
  
public class test {  
 public static Complex add(Complex c1, Complex c2) {  
 Complex temp = new Complex(c1.x+c2.x, c1.y+ c2.y);  
 return temp;  
 }  
  
 public static Integer add(Integer a1, Integer b1) {  
 return a1+b1;  
 }  
  
 public static void main(String[] args) {  
 Complex obj1 = new Complex(4,9);  
 Complex obj2 = new Complex(1,7);  
 Complex t = *add*(obj1, obj2);  
 t.display();  
 System.*out*.println(*add*(5,6));  
 }  
}

Output –

Real: 5, Imaginary: 16

11

Q6. What will be output of below code?

class Complex{  
 int x, y;  
 Complex(int a, int b) {  
 x = a;  
 y = b;  
 }  
 public void display() {  
 System.*out*.println("Real: " + x + ", Imaginary: " + y) ;  
 }  
}  
public class test {  
 public static void main(String[] args) {  
 Complex []objs = new Complex[4];  
 for(int i=0; i < 4; i++) {  
 objs[i] = new Complex(i\*2, i\*5);  
 objs[i].display();  
 }  
 }  
}

Output –

Real: 0, Imaginary: 0

Real: 2, Imaginary: 5

Real: 4, Imaginary: 10

Real: 6, Imaginary: 15

Q7. What will be output of below code?

import java.util.Random;  
import java.lang.Math;  
  
public class test {  
 public static void main(String[] args) {  
 for(int i=0; i < 5; i++) {  
 Random random=new Random();  
 int r = random.nextInt(100);  
 double si = Math.*sin*(r);  
 double co = Math.*cos*(r);  
 System.*out*.println(r + ", " + (si\*si+co\*co));  
 }  
 }  
}

Output –

19, 0.9999999999999999

63, 1.0

60, 1.0

50, 1.0000000000000002

24, 1.0

Q8. What is output of below code?

class Animal{  
 int age;  
 Animal() {  
 age = 20;  
 }  
 public void display() {  
 System.*out*.println(age);  
 }  
}  
  
class Mammal extends Animal{  
 Mammal() {  
 age = 30;  
 }  
}  
  
class Cow extends Mammal{  
 Cow() {  
 age = 40;  
 }  
}  
  
public class test {  
 public static void main(String[] args) {  
 Cow ob = new Cow();  
 ob.display();  
 }  
}

Output –

40

Q9. What is output of below code?

class Animal{  
 int age;  
 Animal() {  
 age = 20;  
 }  
 public void display() {  
 System.*out*.println(age);  
 }  
}  
  
class Mammal extends Animal{  
 Mammal() {  
 age = 30;  
 }  
}  
  
class Cow extends Mammal{  
 Cow() {  
 age = 40;  
 }  
 public void display() {  
 super.display();  
 System.*out*.println(this.age);  
 }  
}  
public class test {  
 public static void main(String[] args) {  
 Cow ob = new Cow();  
 ob.display();  
 }  
}

Output –

40

40

Q10. What is output of below code?

abstract class Parent{  
 int par\_val;  
 Parent(int val) {  
 par\_val = val;  
 }  
 protected abstract void display();  
}  
  
public class test extends Parent {  
 int val;  
 test() {  
 this(10);  
 }  
 test(int init) {  
 super(init+2);  
 if(init > 0)  
 this.val = init;  
 }  
 protected void display() {  
 System.*out*.println(val + " " + par\_val);  
 }  
 public static void main(String[] args) {  
 test obj = new test(10);  
 obj.display();  
 Parent obj2 = new Parent(11);  
 obj2.display();  
 }  
}

Output –

java: Parent is abstract; cannot be instantiated