

Q) public class HelloWorld{

public static void main(String []args){

Child obj = new Child(10);

obj.m1();

}

}

class Child extends Parent{

Child(){

System.out.println("default base const");

}

Child(int a){

super();

System.out.println("overloaded base const"+ a);

}

void m1(){

add(2,3); //this.add(2,3);

}

}

class Parent{

Parent(){

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

}

Parent(int a){

this();

System.out.println("Overloaded Parent const"+ a);

}

void add(int a, int b){

System.out.println(a+b);

}

}

A)

Method Overloading

1) Methods needs to be in the same class, same name

2) Parameters should be diff : number of parameters/ order of parameters/ type of parameters

3) Return types do not matter

4) Exceptions: does not matter

5) Access modifiers: does not matter

Q) public class HelloWorld{

public static void main(String []args){

Parent obj= new Parent();

System.out.println(obj.add(3.4f,4));

}

}

class Parent{

void add(int a, int b){

System.out.println(1);

}

float add(float a, int b) throws RuntimeException{

return a+b;

}

void add(int a, float b)throws java.io.IOException{

System.out.println(3);

}

void add(int a, int b, int c){

System.out.println(4);

}

}

Q) public class HelloWorld{

public static void main(String []args){

Parent obj= new Parent();

System.out.println(obj.add(3,4));

}

}

class Parent{

private Object add(int a, int b){

return 1;

}

}

class Child extends Parent{

private Integer add(int a, int b){

return 2;

}

}

Q) public class HelloWorld{

public static void main(String []args){

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

public String add(int a, int b){

return "1";

}

}

class Child extends Parent{

public Integer add(int a, int b){

return 2;

}

}

@Override?

Q) public class HelloWorld{

public static void main(String []args){

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

public String add(int a, int b){

return "1";

}

}

class Child extends Parent{

@Override

public Integer add(int a, int b, int c){

return 2;

}}

Q) public class HelloWorld{

public static void main(String []args){

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

public Object add(int a, int b) throws java.io.IOException{

return "1";

}

}

class Child extends Parent{

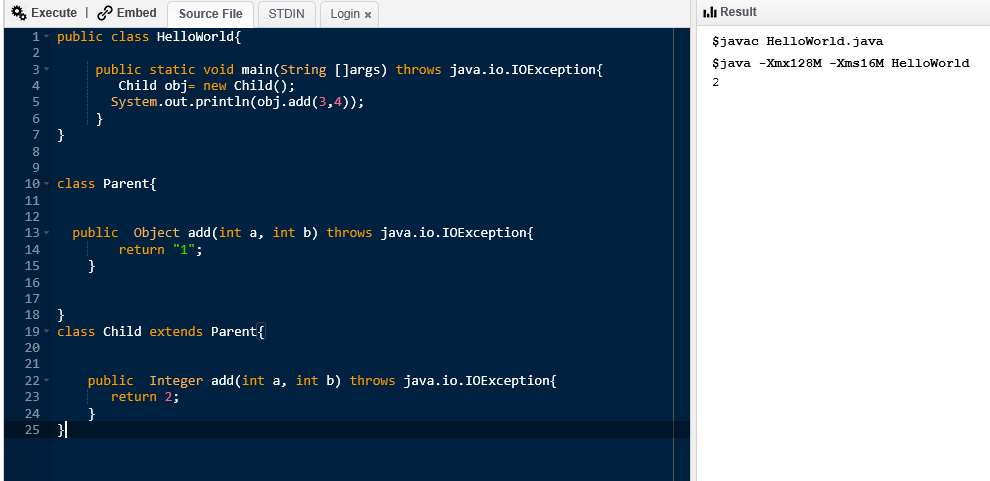
@Override

public Integer add(int a, int b){

return 2;

}

}



Q) public class HelloWorld{

public static void main(String []args) throws java.io.IOException{

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

public Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

public Integer add(int a, int b) throws java.io.IOException{

return 2;

}

}

Q) public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

public Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

public Integer add(int a, int b) throws RuntimeException{

return 2;

}

}

Q) public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

public Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

private Integer add(int a, int b) throws RuntimeException{

return 2;

}

}

Q) public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

private Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

@Override

public Integer add(int a, int b) throws RuntimeException{

return 2;

}

}

Method overriding

1) parent child relationship b/w the classes

2) Parameters: same, method name :same

3) Return types: same or covariant (Parent-child relationship is acceptable)

4) Exceptions : A child class cannot throw any new checked Exception . A parent may throw any exception.

5) Access modifier: Child class method cannot be more restrictive

6) private methods cannot be overridden

7) static methods cannot be overridden

Q) public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

private Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

public Integer add(int a, int b) throws RuntimeException{

return 2;

}

}

Q)

public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

static Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

public Integer add(int a, int b) throws RuntimeException{

return 2;

}

}

Q) public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

static Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

public static Integer add(int a, int b) throws RuntimeException{

return 2;

}

}

Q)

public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

static Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

@Override

public static Integer add(int a, int b) throws RuntimeException{

return 2;

}

}

Q) public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

static Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

public static Integer add(int a, int b) throws RuntimeException{

System.out.println(add(2,3)); //this.add(2,3);

return 2;

}

}

Q)

public class HelloWorld{

public static void main(String []args) {

Child obj= new Child();

System.out.println(obj.add(3,4));

}

}

class Parent{

Parent(){

}

static Object add(int a, int b) {

return "1";

}

}

class Child extends Parent{

Parent(){

}

public static Integer add(int a, int b) throws RuntimeException{

System.out.println(add(2,3)); //this.add(2,3);

return 2;

}

}