# Java: Inheritance

Quiz

# Question 1

```
class A{
A(int i){ this.i=i;}
int i;
}
class B extends A{}
```

Which of the following is true about the code above?

A. The code does not compile because there is no constructor defined in B
B. The code does not compile because no-argument constructor is not defined in A
C. The code does not compile because no-argument constructor is not defined **`in B`**
D. The code compiles fine

# Question 2

```
package a;
public class A{
protected int i;
}}
package b;
public class B extends A{
void f(A a){}
}
```

Which of the following is accessible in the method f()?

A. `this.i`
B. `a.i`
C. `super.i`
D. None of the above.

# Question 3

```
class A {
public static void f(){
System.out.println("fA");   }}
class B extends A{
public void f(){
System.out.println("fB");   }
public static void main(String[] args) {
A a= new B();
a.f();   }}
```

What will happen on compilation or execution of code?

A. fA
B. fB
C. Code will not compile
D. Code will throw runtime error

# Question 4

```
class D{}
class T{
D d;
public Object getD(){ //line 1
return new D(); //line 2
}}
class S extends T{
public D getD(){ // line 3
return new D();
}}
```

Identify the problems in the code?

A. Code will have compilation error at line 1
B. Code will have compilation error at line 2
C. Code will have compilation error at line 3
D. Code will compile clean

# Question 5

```
abstract class A{
static void addUp(int x, int y){
System.out.println(x+y);}
public static void main(String[] a){
A.addUp(5, 10);
}}
```

What will happen on compilation or execution of code?

A. Code does not compile because abstract class must have at least one abstract method
B. Code does not compile because abstract class cannot have static method
C. Code does not compile because abstract class cannot have be instantiated.
D. Code prints 15 on execution

# Question 6

```
class T extends String{
public static void main(String[] a){
String t = new T();
System.out.println(t);
}}
```

What is the result of compilation and execution of the code?

A. Code does not compile because inheritance from **`String`** is prohibited
B. Code does not compile because **`String`** class does not have no-argument constructor
C. Code will compile even if **`String`** is replaced by **`System`**.
D. Code compiles clean

# Question 7

```
class A{
A(){i=0;}
A(int i){this.i=i;}
int i; }
class B extends A{
int j;
B(int j){ this.i=10; this.j=j;}  //line 1
B(){this(5); }
public static void main(String[] a){
A b= new B();
System.out.println(b.i+b.j);//line 2
}}
```

What is the result of compilation/execution of the code?

A. Compilation error at line 1
B. Compilation error at line 2
C. Prints: 15
D. Prints: 5

# Question 8

```
class B {String s1 = "Bs1"; String s2 =
"Bs2";}
class A extends B {
 String s1 = "As1";
 public static void main(String args[])
{
   A x = new A(); B y = (B)x;
   System.out.println(x.s1+" "+x.s2+"
"+y.s1+" "+y.s2);
  }}
```

The code prints

A. Bs1 Bs2 Bs1 Bs2
B. As1 Bs2 As1 Bs2
C. As1 Bs2 Bs1 Bs2
D. As1 As2 Bs1 Bs2

# Question 9

```
class B {
static void main(){System.out.println("main B");}
}
class A extends B {
static void main(){System.out.println("main A");}
public static void main(String args[]) {
    //line 1
    x.main();
}}
```

Which of the following statements in line 1 will print **`main B`**?

**A.** `B x = new A();`

**B.** `A b=(A)new A();`

**C.** `B b=(B)(A)new A();`

**D.** `A b=(A) new B();`

# Question 10

```
package p;
Public class B {
Public Object main(){
System.out.println("main B");
}}
```

The class inheriting from B which is in another package can override **`main()`** method as

A. `public Object main()`
B. `private Object main()`
C. `public String main()`
D. `protected Object main()`

# Question 11

```
class A {
public String toString(){
return "A";
}
class B extends A {
public String toString(String s){
return "B" +s;
}
public static void main(String[] args) {
    B b = new B();
System.out.println(b);
}}
```

Which of the following is true?

**A. Code, on execution prints A**

**B. Code, on execution prints Bnull**

**C. Code, on execution prints Anull**

D. Code does not compile

# Question 12

```
class A {
String s = "A";
private void print(){System.out.print(s);}
public static void main(String[] args) {
A b=new B();
b.print();
 }
}
class B extends A {
String s = "B";
public  void print(){System.out.print(s);    }
    }
```

What happens on compilation and execution of the code?

A. Prints A

B. Prints B

C. Compilation error because of invalid overriding

D. Runtime error because of ClassCastException

# Question 13

```
class A {
String s = "A";
final private void
 print(){System.out.print(s);}
}
```

Class inheriting from A can have **`print()`** method declaration as

A. **`private void print()`**
B. **`final public void print()`**
C. **`void print()`**
D. None of the above since print method is final and cannot be overridden

# Question 14

```
class Tree {
Tree getInstance() { return new Tree();}
}
class Fruit extends Tree {
    //line 1
}
class Mango extends Fruit{    }
```

Which statement(s), inserted at line 1, will NOT compile?

A. `Fruit getInstance() { return this;}`

B. `Mango getInstance() { return this;}`

C. `Tree getInstance() { return this;}`

D. `Object getInstance() { return this;}`

# Question 15

```
1. class Tree {
2. int leaves;
3. @Override
4. public boolean equals(Object o){
5. if(leaves==(Tree)o.leaves)
6. return true;
7. else return false;
8. }}
```

What are the problems with the code listed above

A. There is a warning by compiler for incorrect **`equals`** method
B. There is a compilation error because of incorrect overriding of **`equals`** method
C. A compilation error occurs at Line 5
D. There is no problem with the code

# Question 16

```
class B {
  void f(){ System.out.print("fB");}
  B(){f(); }
}
public class A extends B{
  void f(){ System.out.print("fA");}
A(){  f();}
public static void main(String[] args) {
              new A();
}}
```

Code prints

**A. fBfB**

**B. fAfA**

**C. fAfB**

**D. fBfA**

# Question 17

Which of the following statements are true about the default implementation of the **`public int hashCode()`** method of the **`Object`** class?

A. Every class that overrides **`equals`** methods must overrride **`hashCode`** method also

B. As far as it may be practically possible, the **`hashCode`** method defined by the **`Object`** class does return distinct integers for distinct object.

C. For 2 objects references referring to the same object, the **`hashCode`** method returns the same integer.

D. It returns a fixed number that internally represents the **`Object`** class of the JVM