Polymorphism

QUIZ

```
class A
{
void m1(){System.out.println("A");}
}

class B extends A
{
void m1(){System.out.println("B");}
}

public class Number
{
public static void main(String[] args) {
A ob=new B();
ob.m1();
}
}
```

```
class A
{
}

class B extends A
{
void m1(){System.out.println("B");}
}

public class Number
{
public static void main(String[] args) {
A ob=new B();
ob.m1();
}
}
```

```
class A
{
static void m1(){System.out.println("A");}
}

class B extends A
{
static void m1(){System.out.println("B");}
}

public class Number
{

public static void main(String[] args) {

A ob=new B();
ob.m1();
}
}
```

```
class A
{
void m1(){System.out.println("A");}
void m2(){System.out.println("A m2");}
}

class B extends A
{
 void m1(){System.out.println("B");}
}

public class Number
{
public static void main(String[] args) {
A ob=new B();
ob.m2();
}
}
```

```
class A
{
void m1(){System.out.println("A");}

}

class B extends A
{
 void m1(){System.out.println("B");}
 void m2(){System.out.println("A m2");}
}

public class Number
{
public static void main(String[] args) {
A ob=new B();
ob.m2();     // replace with ((B)ob).m2();
}
}
```

```
class A
{
void m1(){System.out.println("A");}

}

class B extends A
{
 void m1(){System.out.println("B");}
 void m2(){System.out.println("B m2");}
}

public class Number
{
public static void main(String[] args) {
B ob=new A();   // not valid
ob.m1();
}
}
```

```
class A
{
void m1(){System.out.println("A");}
}
class B extends A
{
 void m1(){System.out.println("B");}
 void m2(){System.out.println("B m2");}
}
public class Number
{
public static void main(String[] args) {

B ob=(B)new A();
ob.m1();
}
}
```

```
class A
{
static void m1()
{
System.out.println("test A");
}
}

class B extends A
{
static void m1()
{
System.out.println("test B");
}
}
public class TestDemo {

public static void main(String[] args) {
A b1=new B();
b1.m1();
}
}
```

Given the following,

```
1. import java.awt.*;
2. class Ticker extends Component {
3. public static void main (String [] args) {
4. Ticker t = new Ticker();
5.
6. }
7. }
```

which two of the following statements, inserted independently, could legally be inserted into line 5 of this code? (Choose two.)

A. boolean test = (Component instanceof t);

B. boolean test = (t instanceof Ticker);

C. boolean test = t.instanceof(Ticker);

D. boolean test = (t instanceof Component);

E. boolean test = t.instanceof(Object);

F. boolean test = (t instanceof String);

Given the following,

```
class Foo {
String doStuff(int x) { return "hello"; }
}
```

which method would not be legal in a subclass of Foo?

A. String doStuff(int x) { return "hello"; }

B. int doStuff(int x) { return 42; }

C. public String doStuff(int x) { return "Hello"; }

D. protected String doStuff(int x) { return "Hello"; }

E. String doStuff(String s) { return "Hello"; }

F. int doStuff(String s) { return 42; }

Given the following,

```
1. class Over {
2. int doStuff(int a, float b) {
3. return 7;
4. }
5. }
6.
7. class Over2 extends Over {
8. // insert code here
9. }
```

which two methods, if inserted independently at line 8, will not compile? (Choose two.)

A. public int doStuff(int x, float y) { return 4; }

B. protected int doStuff(int x, float y) {return 4; }

C. private int doStuff(int x, float y) {return 4; }

D. private int doStuff(int x, double y) { return 4; }

E. long doStuff(int x, float y) { return 4; }

F. int doStuff(float x, int y) { return 4; }

Which of these modifiers/specifiers are valid for an abstract class?

A final

B static

C private

D protected

E no specifier

```
1. abstract class A {
2. abstract short m1() ;
3. short m2() { return (short) 420; }
4. }
5.
6. abstract class B extends A {
7. // missing code ?
8. short m1() { return (short) 42; }
9. }
```

which three of the following statements are true? (Choose three.)

A. The code will compile with no changes.

B. Class B must either make an abstract declaration of method m2() or implement method m2() to allow the code to compile.

C. It is legal, but not required, for class B to either make an abstract declaration of method m2() or implement method m2() for the code to compile.

D. As long as line 8 exists, class A must declare method m1() in some way.

E. If line 6 were replaced with ‘class B extends A {‘ the code would compile.

F. If class A was not abstract and method m1() on line 2 was implemented, the code would not compile.

Which two of the following are legal declarations for nonnested classes and interfaces?

(Choose two.)

A. final abstract class Test {}

B. public static interface Test {}

C. final public class Test {}

D. protected abstract class Test {}

E. protected interface Test {}

F. abstract public class Test {}

Which two are valid declarations within an interface? (Choose two.)

A. public static short stop = 23;

B. protected short stop = 23;

C. transient short stop = 23;

D. final void madness(short stop);

E. public Boolean madness(long bow);

F. static char madness(double duty);

Given the following,

```
1. interface DoMath {
2. double getArea(int rad); }
3.
4. interface MathPlus {
5. double getVol(int b, int h); }
6.
7.
8.
```

which two code fragments inserted at lines 7 and 8 will compile? (Choose two.)

A.
```
class AllMath extends DoMath {
double getArea(int r); }
```

B.
```
interface AllMath implements MathPlus {
double getVol(int x, int y); }
```

C.
```
interface AllMath extends DoMath {
float getAvg(int h, int l); }
```

D.
```
class AllMath implements MathPlus {
double getArea(int rad); }
```

E.
```
abstract class AllMath implements DoMath, MathPlus {
public double getArea(int rad) { return rad * rad * 3.14; } }
```

Which three are valid method signatures in an interface? (Choose three.)

A. private int getArea();

B. public float getVol(float x);

C. public void main(String [] args);

D. public static void main(String [] args);

E. boolean setFlag(Boolean [] test []);

Given the following,

1. interface Base {
2. boolean m1 ();
3. byte m2(short s);
4. }

which two code fragments will compile? (Choose two.)

A. interface Base2 implements Base {}

B. abstract class Class2 extends Base {
public boolean m1() { return true; } }

C. abstract class Class2 implements Base { }

D. abstract class Class2 implements Base {
public boolean m1() { return (7 > 4); } }

E. class Class2 implements Base {
boolean m1() { return false; }
byte m2(short s) { return 42; } }