1. C - Legal at compile and runtime

2. A - Illegal both compile and runtime, you cannot assign an object to a sibling reference.

```
parent.java:18: Incompatible type for =. Can't convert DerivedTwo to DerivedOne.

    d1 = d2;
```

3. A - Illegal both compile and runtime, you cannot assign an object to a sibling reference, even with casting.

```
parent.java:21: Invalid cast from DerivedTwo to DerivedOne.
```

4. B - Legal at compile time, but fails at runtime, if the object contained in p is not actually a DeriveOne object, the assignment will cause Java to throw a CastClassException.

```
java.lang.ClassCastException:

        at Test.main(parent.java:17)
```

5. C - Legal at compile and runtime, you can assign an parent object to a subclass with or without casting.(see question 1.)

6. A - RandomAccessStream, the RandomAccessFile does create a new file when it not exists but does not append directly to the end of the file, but you have to set the filepointer to the end of the file. The DataOutputStream by creates a new empty file.

7. B - x=012, the octal number conforms to the type int, the rest needs an explicit cast to the int type to convert. At compile time it will generate the following errors:

```
parent.java:61: Incompatible type for declaration. Explicit cast needed to conve

rt double to int.

    int x=10.0;

parent.java:65: Incompatible type for declaration. Explicit cast needed to conve

rt float to int.

    int x=10f;

parent.java:67: Incompatible type for declaration. Explicit cast needed to conve

rt long to int.

    int x=10L;
```

8. B - break outer, the break at middle will only break the inner loop not the outer loop. The continue statement are to revive a loop not to end.

*9. B - Menu, only components can be added to the container, container itself is a component, button and component are component. Surprise...even an applet is a component. But menu extends MenuComponent which extends Object so it is not a component.*To test:

```
Frame f = new Frame("My"); f.resize(400,500);

Applet a = new Applet(); a.resize(300,50); a.setBackground( Color.red );

f.add( a ); f.show();
```

10. A - the event is handled by the component and doesnot need to be handled by the parent component. The API package documentation has the following comment in java.awt.Component on handleEvent().

Returns true if the event is handled and should not be passed to the parent of this component.

11. A - the event target is the object where the event came from.

12. public static final char CODE='\u30a0';

*13. A, B, C & D, why? This is a tricky question. Both B and D answers are comparing the content of string a,b and stringbuffer d,c but both A and C answers are comparing the object reference number not the content "Hello".*For example if we change the code to this:

```
  a = new String("Hello");

  b = new String("Hello");
```

Then the b==a wouldn't give true because the reference numbers are different.

14. C & E, see compiler output,

```
Loop.java:53: Invalid expression statement.

  int %fred;

      ^

Loop.java:55: Invalid expression statement.

  int *fred;

      ^

Loop.java:57: Invalid expression statement.

  int 2fred;

  ^
```

*15. B & D, after closing a class the interpreter can only expect a class of interface declaration. But only if the classes and/or interfaces defined are **not public!!!** See the compiler output for the code:*

```
X.java:7: Identifier expected.

abstract void met();

        ^

X.java:7: 'class' or 'interface' keyword expected.

abstract void met();

         ^

X.java:13: Class or interface declaration expected.

package java.util.*;

^
```

16. A, int count = args.length, the length will not include the command java only parameters that follow after the command.

```
So the command:

java Test and or

will give args.length = 1;

args[0] = "and"

args[1] = "or"
```

17. B, an filteroutputstream can only be build with an OutputStream. See API documentation the constructor for filteroutputstream:

```
 FilterOutputStream(OutputStream)

Creates an output stream filter.
```

18. C & E, an AWTException is extended from Exception therefor the method throws either AWTException or Exception.

19. A & C, a Textfield is at construction always editable and a TextField constructor with an integer has the following description. See API documentation java.awt.TextField:

```
public TextField(int cols)

Constructs a new TextField initialized with the specified columns.
```

20. this(a);

20. this(a);

21. C,D and E.

22. C, for details see java -cs

23. A and E, (no comment sorry)

```
java.awt.Event[id=1001,x=9,y=101,target=java.awt.TextField[textfield0,9,101,234x
23,text=,editable,selection=0-0],arg=]
java.awt.Event[id=1001,x=363,y=101,target=java.awt.Button[button0,363,101,27x23,
label=ok],arg=ok]
```

24. Causes a thread to wait forever until it is notified. (see API java.lang.Object)

25. B & D, the SubClass doesnot implement the same constructor as SuperClass and the SubClass doesnot expect any constructor argument (wrong number of arguments)

26. public native void nativeHelloWorld();

27. A, C, D and E.