1. Dado o seguinte código :

```
interface Simples {
        boolean mostra( );
        byte ver(short s);
}
```

Qual fragmento de código irá compilar? (Marque todas as corretas)

A. interface Simples2 implements Simples { }

B. abstract class Class2 extends Simples {
public boolean mostra( ) { return true; } }

C. abstract class Class2 implements Simples { }

D. abstract class Class2 implements Simples {
public boolean mostra( ) { return (true) ; } }

E. class Class2 implements Simples {
boolean mostra( ) {return false; }
byte ver(short s) { return 42; } }

{Estão corretas **C** e **D**. C está correto porque uma classe abstrata não precisa implementar nenhum dos métodos de sua interface. D está correto porque o método foi implementado corretamente.
A está incorreta porque uma interface não pode implementar outra.
B está incorreta porque uma classe não pode herdar uma interface.
E está incorreta porque todos os métodos de uma interface são públicos, logo, os métodos que os implementam também devem ser.}

2. Qual das seguintes opções declara uma classe abstract compilável? (Marque todas as Corretas.)

A. public abstract class Canine { public String speak( ); }

B. public abstract class Canine { public String speak ( ) { } }

C. public class Canine { public abstract String speak ( ) ; }

D. public class Canine abstract { public abstract String speak ( ) ; }

{B está correta.
A está incorreta porque o método declarado não é abstract.
C está incorreta porque a classe não foi declarada como abstract e por isso não pode conter métodos abstract.
D está incorreta porque a key word abstract está após o nome da classe.}

3. Qual afirmativa está correta?

A. “X estende Y” é correto se, e somente se, X for uma classe e Y for uma interface.

B. “X estende Y” é correto se, e somente se, X for uma interface e Y for uma classe.

C. "X estende Y" é correto se X e Y forem ambos classes ou ambos interfaces.

D. "X estende Y" é correto para todas as combinações de X e Y sendo classes e/ou interfaces.

{C está correta.
Classe estende classe, interface estende interface.}

4. Quais das seguintes são declarações válidas? (Marque todas as corretas.)

A. int $x;

B. int 123;

C. int _123;

D. int #dim;

E. int *divide;

F. int central_sales_region_Summer_200S_gross_sa1es

{Estão corretas A, C e F}

5. Quais das seguintes opções são declarações válidas? (Marque todas as corretas.)

A. short x [ ];
B. short [ ] y;
C. short [5] x2;
D. short z2 [5] ;
E. short [ ] z [ ] [ ];
F. short [ ] y2 = [5] ;

{Estão corretas A, B e E.
C e D estão erradas porque não se pode especificar a quantidade de elementos na declaração do vetor.
F está incorreto na forma utilizada para inicializar o vetor.}

6. Quais afirmativas são verdadeiras? (Marque todas as corretas)

A. Os relacionamentos Tem-Um sempre dependem da herança.
B. Os relacionamento Tem-Um sempre dependem das variáveis de instâncias.
C. Os relacionamento Tem-Um sempre precisam de pelo menos dois tipos de classes.
D. Os relacionamento Tem-Um sempre dependem do polimorfismo.

{B está correta.}

7. Usando os fragmentos abaixo, complete o seguinte código de forma que ele compile. Observação, talvez não seja preciso preencher todos os espaços em branco.

Código:

```
class Agedp{
__________  __________  __________  __________  __________
  public Agedp(int x) {
        __________  __________  __________  __________  __________
  }
}

public class Kinder extends Agedp {
__________  __________  __________  __________  __________  __________
  public Kinder (int x) {
        __________  __________  __________  __________  __________ ( );
  }
}
```

Fragmentos: Use cada um dos seguintes fragmentos quantas vezes for preciso, ou deixe sem uso:

Agedp	super	this	
(	)	{	}
;			

{Partindo do princípio que o construtor de Kinder não poderá ficar vazio, pois já há o () no final da linha, teríamos que colocar um super ou this. Se fossemos utilizar o this, teríamos que construir outro construtor para Kinder e isso não é possível utilizando os fragmentos. Assim, optamos pelo super. O super vazio precisa de um construtor vazio em Agedp, o que é facilmente construído utilizando os fragmentos disponibilizados. Os demais espaços permanecem em branco.}

```
class Agedp{
  public Agedp () { }
  public Agedp(int x) {
          _____   _____   _____   _____   _____
  }
}

public class Kinder extends Agedp {
_____   _____   _____   _____   _____   _____
  public Kinder (int x) {
          super ( );
  }
}
```

7. Enumere as 2 e 3 colunas de acordo com a primeira:

(1) boolean	( ) Depende da JVM	( ) lógico
(2) char	( ) 8 bits	(, , ,) inteiro
(3) int	(,) 16 bits	(,) real
(4) byte	(,) 32 bits	( ) caracter
(5) double	(,) 64 bits	
(6) long		
(7) short		
(8) float		

{

(1) boolean	(1) Depende da JVM	(1) lógico
(2) char	(4) 8 bits	(3 , 4 , 6 , 7) inteiro
(3) int	(2 , 7) 16 bits	(5 , 8) real
(4) byte	(3 , 8) 32 bits	(2) caracter
(5) double	(5 , 6) 64 bits	
(6) long		
(7) short		
(8) float		

}