# Java Bronze　後期中間試験　令和1年12月19日　実施

## 従業員の情報を表すEmployeeくらすがある。正しくカプセル化されるようにこのクラスを修正したい。修正内容として、正しいものを2つ選びなさい。 （　C,D　 ）

|  |
| --- |
| public class Employee{  String corporateName;  String corporateAddress;  int employeeNo;  String name;  } |

1. すべてのフィールドのアクセス修飾子をprivateにする
2. フィールドにアクセするためのgetter、setterメソッドを追加する
3. corporateNameとcorporateAddressフィールドを他のクラスに移動する
4. 名前を名乗ってあいさつするメソッドを追加する
5. 給与計算のメソッドを追加する

## 次のコードをコンパイル、実行した時の結果として、正しいものを選びなさい。 （　E　 ）

|  |
| --- |
| public class Main {  public static void main(String[] args) {  for (int i = 0;; i++) {  int j = 0;  while (j <= 3)  System.out.println(j++);  }  }  } |

1. 「123」と1回表示される
2. 「0123」と1回表示される
3. 「1234」と1回表示される
4. 「1234」が無限に表示される
5. 「0123」が無限に表示される
6. コンパイルエラーとなる

## クラスの宣言として有効なものを3つ選びなさい （　BCD　 ）

1. public class Test extends java.lang.\*;
2. public class Test extends java.lang.Object { }
3. final class Test Test { }
4. public class Test { }
5. public class Test implements Object { }

## 次のプログラムの6行目「//insert code here」に入るコードとして、正しいものを選びなさい。 （　B　 ）

|  |
| --- |
| public class Sample {  private String name;  private int price;  public String TMP = "sample";  public Sample() {  // insert code here  }  public Sample(String name) {  this.name = name;  }  public void print() {  System.out.println(name + ":" + price);  }  public static void main(String[] args) {  Sample s = new Sample();  s.print();  }  } |

1. this.price = 100;  
   this( "sample" );
2. this( "sample" );  
   this.price = 100;
3. this( TMP );  
   this.price = 100;
4. this.price = 100;  
   this( TMP );
5. Sample( "samle");  
   this.price = 100;

## データ隠蔽を実現するためには、フィールドをどのように修飾すればよいか。正しいものを1つ選びなさい。 （　C　 ）

1. public final
2. public static
3. private
4. private static
5. private final

## 次のコードをコンパイル、実行した時の結果として、正しいものを選びなさい。 （　D　 ）

|  |
| --- |
| public class Main {  public static void main(String[] args) {  int x = 6;  int y = x++;  int z = ++y;  System.out.println(x + " " + y + " " + z);  }  } |

1. 「」と表示される
2. 「」と表示される
3. 「」と表示される
4. 「」と表示される
5. コンパイルエラーとなる

## 次のコードをコンパイル、実行した時の結果として、正しいものを選びなさい。 （　D　 ）

|  |
| --- |
| public class Main {  public static void main(String[] args) {  for (int i = 0; ++i < 5;)  System.out.println(i + " ");  }  } |

1. 何も表示されない。
2. 「01234」と表示される
3. 「012345」と表示される
4. 「」と表示される
5. 「」と表示される
6. コンパイルエラーになる

## 次のインターフェースのコンパイルを成功させるには、どのコードを2行目の「//insert code here」に入れればよいか。正しいものを選びなさい。 （　A,D　 ）

|  |
| --- |
| public interface Sample{  // insert code here  } |

1. public void setVal( String val );
2. private void setVal( String val );
3. String val;
4. void setVal( String val);
5. public static void setVal( String val);

## 次のプログラムをコンパイル、実行した時の結果として、正しいものを選びなさい。 （　B　 ）

|  |
| --- |
| public class A{  public void test(){  System.out.println( "A");  }  } |

|  |
| --- |
| public class B{  public void test(){  System.out.println( "B");  }  } |

|  |
| --- |
| public class Main{  public static void main( String [ ] args ){  A a = new B();  a.test();  }  } |

## カプセル化の維持に欠かせない原則として、正しいものを1つ選びなさい。 （　A　 ）

1. データ隠蔽
2. 情報隠蔽
3. 抽象化
4. データ抽象

## データ隠蔽の説明として、最も適切なものを1つえらびなさい。 （　B　 ）

1. 関係するデータとそのデータを扱う処理をひとまとめにする
2. 属性の公開範囲を制限する
3. getterやsetterなどのアクセサメソッドを提供する
4. 実装の詳細を隠蔽する

## カプセル化の説明として、正しいものを一つ選びなさい。 （　C　 ）

1. 同種の異なるインスタンスを同じ型として扱う
2. 公開すべきものと非公開にすうべきものを区別して扱う
3. 関係するものをひとまとめにする
4. クラスのフィールドを隠蔽し、他のクラスからは直接使えないようにする。

## 次のコードをコンパイル、実行した時の結果として、正しいものを1つ選びなさい。 （　D　 ）

|  |
| --- |
| public class Main{  public static void main( String[] args){  int i = 5;  System.out.println( ( i += 5) + ":" + ( i--));  }  } |

1. 「5:5」と表示される
2. 「5:4」と表示される
3. 「5:9」と表示される
4. 「10:10」と表示される
5. 「10:9」と表示される

## 共通部分だけを抽出し、それ以外を無視して扱うことを何と呼ぶか。正しい用語を選びなさい。 （　A　 ）

1. 抽象化
2. カプセル化
3. データ隠蔽
4. 情報隠蔽

## 次のSampleクラスを継承したサブクラスを定義するときに、サブクラスに定義したメソッドのうち、Sampleクラスのメソッドを正しくオーバーライドしているものを選びなさい。(2つ選択） （　A、D　 ）

|  |
| --- |
| public class Sample{  void methodA(){}  void methodB(int a ){}  void methodC( int a, int b){  return 0;  }  voiod methodD( int a ){  return 1;  }  } |

1. public void methodA() {}
2. public void methodB( long a ) {}
3. public int methodC( char a , int b ) { return 0 ; }
4. public int methodD( int i ){ return 1 ; }

## 次のコードをコンパイル、実行した時の結果として、正しいものを選びなさい。 （　B　 ）

|  |
| --- |
| public class Main {  public static void main(String[] args) {  int a = 12;  int b = 8;  if ( a >= 10 || b >= 10){  a /= 2;  b += a;  }else  a /=2;  b += a;  System.out.println(a + ":" + b);  }  } |

1. 「6：14」と表示される
2. 「6：20」と表示される
3. 「12：20」と表示される
4. コンパイルエラーになる

## 以下の中から、情報隠蔽にかかわりの深いキーワードを選びなさい。 （　ADE　 ）

1. ポリモーフィズム
2. カプセル化
3. アクセサメソッド
4. パッケージ
5. インタフェース
6. インスタンス化

## 以下の中から、アクセス修飾子privateで就職できるものを選びなさい。 （　B, C, F　 ）

1. インタフェースのフィールド
2. クラスのコンストラクタ
3. クラスのフィールド
4. クラスの抽象メソッド
5. インタフェースのメソッド
6. クラスの具象メソッド

## ある企業はGUIベースのアプリケーション開発を希望しており、将来的な拡張として、WEBベースのアプリケーションへの移行を予定している。このアプリケーションを作成するには、どのJavaテクノロジを使用するのが良いか。正しいものを一つ選びなさい。 （　A　 ）

1. Java　SE
2. Java　EE
3. Java　ME
4. Java　DB

## Javaに関する説明として、正しいものを３つ選びなさい。 （　CDF　 ）

1. プラットフォームに依存する
2. 単一スレッドのアプリケーションのみをサポートする
3. 自己メモリ管理をサポートする
4. アーキテクチャに依存しない。
5. プログラマーは、メモリを直接操作できる
6. 実行時にコンパイルされる

## 次のコードをコンパイル、実行した時の結果として、正しいものを選びなさい。 （　B　 ）

|  |
| --- |
| public class Bridge {  String name;  public static void main(String[] args) {  Bridge b = new Bridge();  if (b.name == "")  b.name = "Brooklyn";  System.out.println(b.name);  }  } |

1. 何も表示されない
2. 「null」と表示される
3. 「Brooklyn」と表示される
4. コンパイルエラーになる
5. 実行時に例外が発生する

## クラス定義の記述として、正しいものを１つ選びなさい。 （　　 ）

1. class ClassA ( int a ){  
    // クラス内定義  
   }
2. ClassA {  
    // クラス内定義  
   }
3. class ClassA ( ) {  
    // クラス内定義  
   }
4. class ClassA {  
    // クラス内定義  
   }

## xxxというパッケージにアクセスできるMainというクラスを、xxx.hogeというパッケージ内に作成するには、どのようにクラスを宣言すればよいか。正しいものを選びなさい。 （　B　 ）

1. import xxx.\*\*\*;  
   package xxx.hoge;

public class Main {

// any code

}

1. package xxx.hoge;  
   import xxx.\*;  
   public class Main{  
    // any code  
   }
2. import xxx;

package xxx.hoge;  
public class Main {  
 // any code   
}

1. package xxx.hoge ;  
   import xxx;

public class Main{

// any code   
}

1. package xxx.\*;  
   public class Main {  
    // any code   
   }

## 次のコードをコンパイル、実行した時の結果として、正しいものを選びなさい。（１つ選択） （　D　 ）

|  |
| --- |
| public class Train{  static String name = "none";  public Train( String name){  this.name = name;  }  public static void main( String [] args){  Train t1 = new Train();  Train t2 = new Train( "aline");  System.out.println( t1.name + " " + t2.name );  }  } |

1. ｢none aline｣と表示される
2. 「null aline」と表示される
3. 「aline aline」と表示される
4. コンパイルエラーになる
5. 実行時に例外が発生する

## 次のプログラムをコンパイル、実行した時の結果として、正しいものを１つ選びなさい。 （　A　 ）

|  |
| --- |
| public interface Sample{  void test();  } |

|  |
| --- |
| public class A implements Sample{  public void test(){  System.out.println( "A");  }  } |

|  |
| --- |
| public class B extends A {  public void test(){  System.out.println( "B");  }  } |

|  |
| --- |
| public class Main{  public static void main( String[] args){  Sample[] samples = { new A(), new B()};  for ( Sample s : samples ){  s.test();  }  }  } |

1. 「A」「B」の順に表示される
2. 「B」「A」の順に表示される
3. 「A」「A」の順に表示される
4. 「B」「B」の順に表示される
5. Bクラスでコンパイルエラーが発生する
6. Mainクラスでコンパイルエラーが発生する