理解度テスト　解説

1、 E

問題　配列の宣言・生成方法として正しいものを選びなさい。(1つ選択)

- ☐ A.int[] array = new int[];
- ☐ B.int() array = new int(3);
- ☐ C.int array = new int[3];
- ☐ D.int[3] array = new int[];
- ☑ E.int[] array = new int[3];

配列に関する問題です。
配列型変数は参照型変数の一種です。
配列変数は配列への参照を保持します。
配列は、指定された個数の要素を持ち、各要素にデータを保持できます。
要素を指定するための添字は0から開始されます。
1からではないことに注意しましょう。
たとえば、配列の長さが3の場合、添字は、0、1、2を指定できます。
配列を利用するためには、配列型変数の宣言と配列の格納領域の生成を行う必要があります。
配列型変数の宣言は、次のように記述します。

書式　要素のデータ型[] 変数名;

配列の格納領域の生成は、newキーワードを用いて次のように記述します。

書式　変数名 = new 要素のデータ型[要素数];

2、 D

問題　配列の変数名をarrayとした場合、配列の長さを参照する方法として正しいものを選びなさい。(1つ選択)

- ☐ A.array.size;
- ☐ B.array.length();
- ☐ C.array length;
- ☑ D.array.length;
- ☐ E.array.size();

配列の要素数を調べる方法に関する問題です。
配列の要素数を参照するには、次のように記述します。

書式　配列の変数名.length

したがって、選択肢Dが正解です。

lengthは配列が確保した要素の数を数えます。

値が入っている要素を返すわけではありません。
たとえば、次のコードは配列に要素を入れるかどうかにかかわらず、確保した要素数である「3」を表示します。

例：lengthの使用例

```
Object[] array = new Object[3];
System.out.println(array.length);
```

3、 C

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

```
1 | public class Main {
2 |     public static void main(String[] args) {
3 |         int a = 0;
4 |         if (a = 0) {
5 |             System.out.print("A");
6 |         }
7 |     }
8 | }
```

☐ A.「A」と表示される。
☐ B.何も表示されない。
☑ C.コンパイルエラーになる。

if文の文法に関する問題です。
条件によって処理を分岐したいときには、if文を使います。
if文の書式は次のとおりです。

書式

```
if (条件式) {
    // 処理
}
```

「条件式」の結果は、boolean型の値である必要があります。
条件式の結果が trueの場合、ifブロック内が処理されます。
falseの場合は、ifブロック内は処理されません。
なお、処理を表すコードが1行しかない場合、ifブロックを表す中カッコ「{ }」は省略可能です。
設問のコードの4行目では、if文の条件式は変数aに代入しているだけで boolean型の値を戻しません。そのため、コンパイルエラーになります。

したがって、選択肢Cが正解です。

選択肢Aのように「A」と表示するためには、条件式の結果がtrueになるよう、「a == 0」と記述します。選択肢Bのように何も表示しないようにするためには、条件式の結果がfalseとなるよう、たとえば「a != 0」などと記述します。

4、 B

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

```
1  | public class Main {
2  |     public static void main(String[] args) {
3  |         int a = 0;
4  |         if (a == 1) {
5  |             System.out.print("A");
6  |         } else {
7  |             System.out.print("B");
8  |         }
9  |         System.out.print("C");
10 |     }
11 | }
```

☐ A.「AC」と表示される。
☑ B.「BC」と表示される。
☐ C.「C」と表示される。
☐ D.コンパイルエラーになる。
☐ E.何も表示されない。

if-else文に関する問題です。
条件に合致した場合と、合致しなかった場合で処理を分岐したいときは、 if-else文を使います。
if-else文の書式は、次のとおりです。

書式

```
if (条件式) {
    // 条件式がtrueの場合の処理
} else {
    // 条件式がfalseの場合の処理
}
```

条件式の結果がtrueの場合にはifブロック内が、falseの場合にはelseブロック 内が処理されます。
各ブロック内の処理が1文しかないのであれば、中カッコ「{ }」は省略可能です。
設問のコードでは変数aに0を代入しているため、4行目の条件式はfalseを戻します。
よって、5行目は処理されず、7行目が処理され、「B」がコンソールに出力されます。
if-else文の終了後、9行目が処理され、「C」がコンソールに出力されます。

したがって、選択肢Bが正解です。

5、 C

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

```
1  | public class Main {
2  |     public static void main(String[] args) {
3  |         int a = 80;
4  |         if (a < 50) {
5  |             System.out.print("A");
6  |         } else if (a < 70) {
7  |             System.out.print("B");
8  |         } else {
9  |             System.out.print("C");
10 |         }
11 |     }
12 | }
```

☐ A.「A」と表示される。
☐ B.「B」と表示される。
☑ C.「C」と表示される。
☐ D.コンパイルエラーになる。
☐ E.何も表示されない。

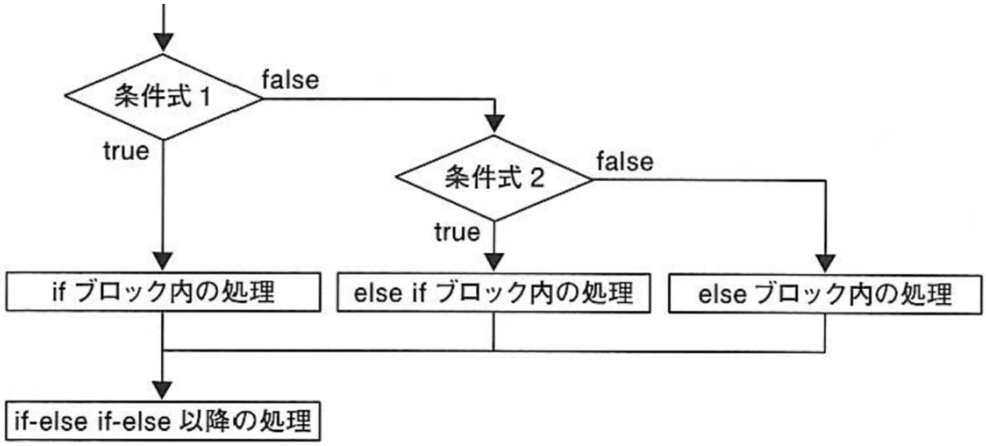
if-else if文に関する問題です。
複数の条件がある場合で、条件ごとに処理内容を分岐したいときには、
if-else ifを使います。

書式

```
if (条件式1) {
    // 条件式1がtrueの場合の処理
} else if (条件式2) {
    // 条件式2がtrueの場合の処理
} else {
    // 条件式1、条件式2がfalseの場合の処理
}
```

条件式1の結果がtrueの場合は、ifブロック内の処理が実行されます。条件式1が
falseの場合は条件式2が評価され、その結果がtrueの場合はelse ifブロック内の
処理が実行されます。もし、条件式2の結果もfalseの場合は、elseブロック内の
処理が実行されます。次の図は、if-else if文の分岐処理を表したものです。

【if-else文】



なお、各ブロック内の処理が1文しかないのであれば、中カッコ「{}」は省略可能です。
また、else if句は、必要に応じて複数にすることもできます。そのほかに、else句を
省略することもできます。そのほかに、else句を省略することもできます。次のコードは、
複数のelse if句があり、else句を省略した例です。この場合、それぞれの条件に
合致しなければ、何も行わないことになります。

例：複数のelse if句とelse句の省略

```
if (a < 50) {
    System.out.print("A");
} else if (a < 70) {
    System.out.print("B");
} else if (a < 80) {
    System.out.print("C");
}
```

設問のコードでは、変数aが80で初期化されています。4行目の条件式はfalseを
戻すため、ifブロック内では処理されません。続いて6行目の条件式が評価され、
falseが戻されるためelse ifブロック内は処理されません。4行目、6行目の条件式の
結果がfalseのため、elseブロック内が処理され、「C」がコンソールに出力されます。

したがって、選択肢Cが正解です。