Home Work.

Week. 7. 21/10/23

1. Rules to declare Constructor.

a. Constructor name & class name must be Same
b. Constructor able to take parameters
c. Constructor not allowed return type

2. Constructor name & class name must be Same
Constructor able to take parameters
Constructor not allowed return type

3. Constructor name & class name must be Same
Constructor able to take parameters
Constructor not allowed return type

4. ~~Constructor not allowed~~
Constructor name & class name must be Same
Constructor able to take parammeters
Constructor not allowed return type.

5. Constructor name & class name must be Same
Constructor able to take parameters
Constructor not allowed retyrn type

2. if Statement Systex.

1. if (condition) {
       if body;
   }

2. if (condition) {
       if body;
   }

3. if (condition) {
       if body;
   }

4. if (condition) {
       if body;
   }

5. if (condition) {
       if body;
   }

3. if-else System

1.
```
if(condition){
    if body; (true body)
}else{
    else body; (false body)
}
```

2.
```
if(condition){
    if body; (true body)
}else{
    else body; (false body)
}
```

3.
```
if(condition){
    if body; (true body)
}else{
    else body; (false body)
}
```

4.
```
if(condition){
    if body; (true body)
    else body; (false body)
}
```

5.
```
if(condition){
    if body; (true body)
    else body; (false body)
}
```

4. Nested if-Else Syntax

1.
```
if (condition){
        if body;
}else if (condition){
        else if body;
}else{
        else body;
}
```

2.
```
if (condition){
        if body;
}else if (condition){
        else if body;
}else{
        else body;
}
```

3.
```
if (condition){
        if body;
}else if (condition){
        else if body;
}else{
        else body;
}
```

4.
```
if (condition){
        if body;
}else if (condition){
        else if body;
}else{
        else body;
}
```

5.
```
if (condition){
        if body;
}else if (condition){
        else if body;
}else{
        else body;
}
```

5. Instance relate to object

- Instance block is used for initializing the instance variable
- This block executed when the object is created
- Instance block is executed before Constructor

- Instance block is used for initializing the instance variable
- This block executed when the object is created
- Instance block is executed before Constructor

- Instance block is used for initializing the instance variable
- This block executed when the object is created
- Instance block is executed before Constructor

- Instance block is used for initializing the instance variable
- This block executed when the object is created
- Instance block is executed before Constructor

- Instance block is used for intiliging the instance variable
- This block executed when the object is created
- Instance block is executed before Constructor

6. Static relate to Class 5 times

1. Static block is executed when class is loaded
   - Instance block is executed when object is before Constructor
   - Constructor is executed when object is created

2. Static block is executed when class is loaded
   - Instance block is executed when object is before Constructor
   - Constructor is executed when object is created

3. Static block is executed when class is loaded
   - Instance block is executed when object is before Constructor
   - Constructor is executed when object is Created

4. Static block is executed when obj class is loaded
   - Instance block is executed when object is before Constructor
   - Constructor is executed when object is Created

5. Static block is executed when class is loaded
   - Instance block is executed when object is before Constructor
   - Constructor is executed when object is created

7. Switch Statement System.

1)
```
Switch (expression){
    case lable1:
        sout();
        break;
    Case lable2:
        sout();
        break;
    Case lable n:
        sout();
        break;
    default:
        sout();
```

2)
```
switch (expression){
    Case lable 1:
        sout();
        break;
    Case lable 2:
        Sout();
        break;
    Case lable n:
        sout();
        break;
    default:
        sout();
```

3)
```
Switch (expression){
    Case lable1:
        Sout();
        break;
    Case lable 2:
        Sout();
        break;
    Case lable n:
        Sout();
        break;
    default:
        sout();
```

4)
```
switch (expression){
    case lable1:
        sout();
        break;
    Case lable 2:
        Sout();
        break;
    Case lable n:
        Sout();
        break;
    defaut:
        Sout();
```

5)
```
switch (expression){
    Case lable1:
        Sout();
        break;
    Case lable2:
        Sout();
        break;
    Case lable n:
        Sout();
        break;
    default:
        Sout();
```

(8) Array - Single and Multi-Dimensional

Syntax

Single dimensional Array.

Approach 1

1. int [ ] a;
   int [ ] b;
   int c [ ];

   int d [ ] = {10, 20, 30, 40};
   int d [ ] = {10, 20, 30, 40};
   int d [ ] = {10, 20, 30, 40};
   int d [ ] = {10, 20, 30, 40};

2. int [ ] a;
   int [ ] b;
   int c [ ];

   int d [ ] = {10, 20, 30, 40};

Approach 2

3. int [ ] a;
   int [ ] b;
   int c [ ];

   ① int [ ] a = new int [100];
   a [0] = 10;
   a [1] = 20;
   a [2] = 30;
   a [3] = 40;

4. int [ ] a;
   int [ ] b;
   int c [ ];

   ② int [ ] a = new int [100];
   a [0] = 10;
   a [1] = 20;
   a [2] = 30;
   a [3] = 40;

5. int [ ] a;
   int [ ] b;
   int c [ ];

③ int [ ] a = new int [100];
a [0] = 10;
a [1] = 20;
a [2] = 30;
a [3] = 40;

④ int [ ] a = new int [100];
a [0] = 10;
a [1] = 20;
a [2] = 30;
a [3] = 40;

⑤ int [ ] a = new int [100];
a [0] = 10;
a [1] = 20;
a [2] = 30;
a [3] = 40;

Two / Multi dimensional

~~int a [] [] = 100;~~

① int a [] [] = new int [3] [2];
a [0] [0] = 100;
a [0] [1] = 200;
a [1] [0] = 300;
a [1] [1] = 400;
a [2] [0] = 500;
a [2] [1] = 600;
sout ("Number of rows:" + a.length);
sout ("Number of columns:" + a[0].length);
for (int i=0; i<a.length; i++) {
for (int j=0; j<a[i].length; j++){
Sout (a[i][j]);
}
}
}

② int a [] [] = new int [3] [2];
a [0] [0] = 100;
a [0] [1] = 200;
a [1] [0] = 300;
a [1] [1] = 400;
a [2] [0] = 500;
a [2] [1] = 600;
Sout ("Number of rows:" + a.length);
Sout ("Number of columns:" + a[0].length);
for (int i=0; i<a.length; i++ {
for (int j=0; j<a[i].length; j++){
Sout (a[i][j]);
}
}
}

③ int a [] [] = new int [3] [2];
a [0] [0] = 100;
a [0] [1] = 200;
a [1] [0] = 300;
a [1] [1] = 400;
a [2] [0] = 500;
a [2] [1] = 600;
Sout ("Number of rows:" + a.length);
Sout ("Number of columns:" + a[0].length);
for (int i=0; i<a.length; i++){
for (int j=0; j<a[i].length; j++){
Sout (a[i][j]);
}
}
}

④ int a [] [] = new int [3] [2];
a [0] [0] = 100;
a [0] [1] = 200;
a [1] [0] = 300;
a [1] [1] = 400;
a [2] [0] = 500;
a [2] [1] = 600;
Sout ("Number of rows:" + a.length);
Sout ("Number of columns:" + a[0].length;
for (int i=0; i<a.length; i++){
for (int j=0; j<a[i].length; j++){
Sout (a[i][j]);
}
}
}

(5) int a[][] = new int [3][2];

```
a[0][0] = 100;
a[0][1] = 200;
a[1][0] = 300;
a[1][1] = 400;
a[2][0] = 500;
a[2][1] = 600;
Sout("Numbe of rows:" + a.length);
Sout("Number of columns:" + a[0].length);
for(int i=0; i<a.length; i++){
for(int j=0; j<a[i].length; j++){
    Sout(a[i][j]);
}
}
```