intro-silly.md 10/3/2022

The Stack Grows Downwards In Memory

```
public static void main(String[] args)
{
  int i=1;
  int a=4;
}
```

- Frames are added to the stack at the highest available address (on the stack)
- Here we illustrate this by showing the stack with the highest address at the top
- The stack fills "downwards"

methods & frames

```
</div>
```

Each method call adds a new frame to the stack

```
public static void main(String[] args)
{
  int i=1;
  int a=4;
```

silly(); }

void silly() { float p=5.6f; }

- When a method is **called** a new frame is added to the stack
- A method can only access its own frame
- Code in silly() cannot use the variables created in main()
- A **frame** limits the **scope** of a variable

methods & frames

```
</div>
```

Frames & Variables

```
public static void main(String[] args)
{
```

intro-silly.md 10/3/2022

```
int i=1;
int a=4;
```

silly(); }

void silly() { float p=5.6f; int i=3; }

• This is why we can use the same variable name in different methods without changing the value everywhere

methods & frames

```
</div>
```

Frames & Scope

```
public static void main(String[] args)
{
  int i=1;
  int a=4;
```

silly(); }

void silly() { float p=5.6f; int i=3; i--; }

• This is why we can use the same variable name in different methods without changing the value everywhere

methods & frames

```
</div>
```

```
public static void main(String[] args)
{
  int i=1;
```

i = silly(i); }

void silly(int locali) { }

• This is why we need to pass values as parameters if we want to use them in the method

```
methods & frames
```

intro-silly.md 10/3/2022

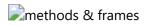
```
</div>
```

```
public static void main(String[] args)
{
  int i=1;
```

```
i = silly(i); }
```

void silly(int locali) { locali = 3; return locali; }

• Similarly if we want to pass back a value to the calling method we have to use a return



</div>