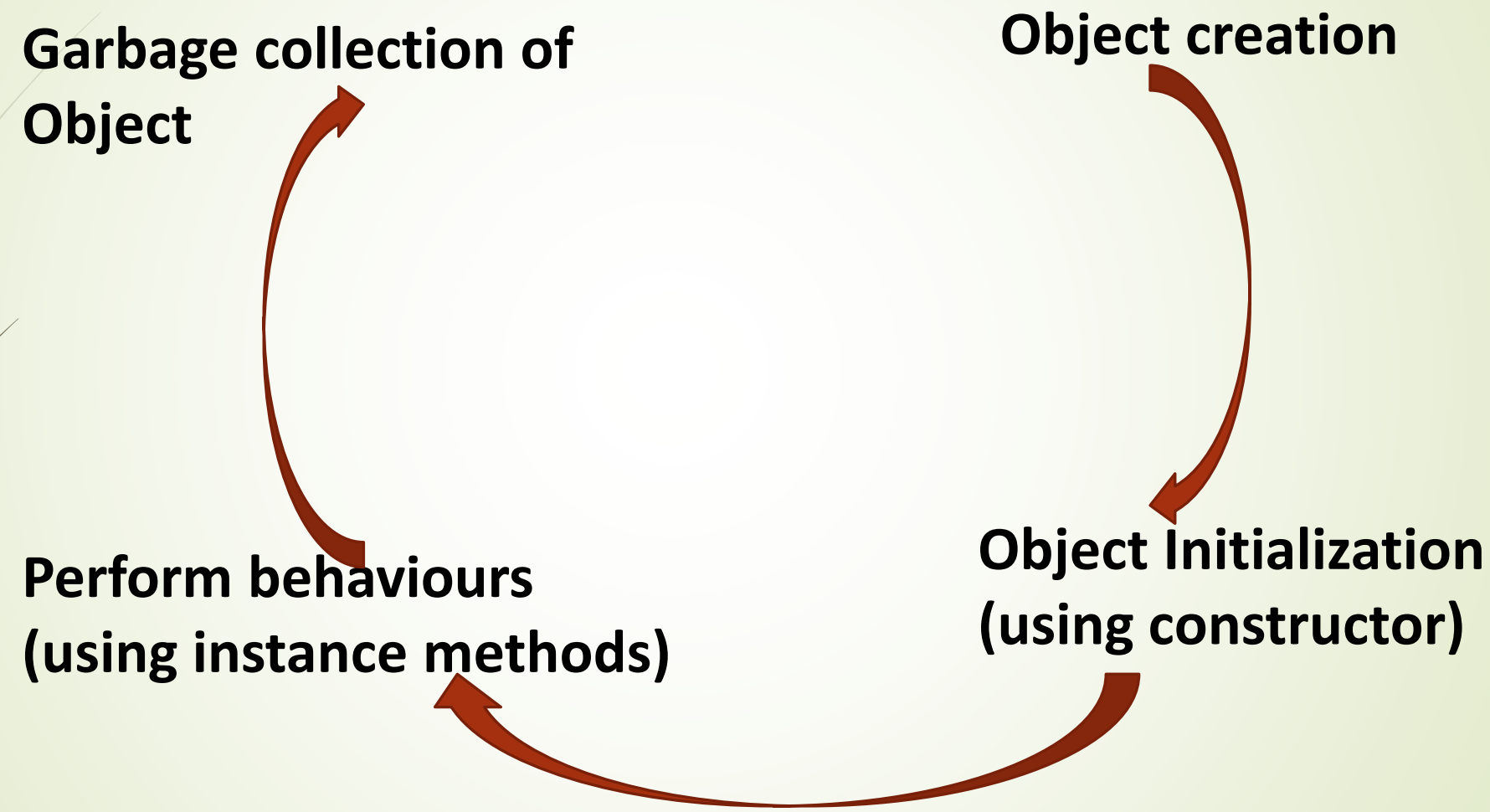
Objects and Reference

Object and Class

- **What is Object?**
  - Any real life entity which has properties and behaviours
  - **Instance of class / variable of class**
- **What is class?**
  - **Blueprint for Object**
  - template that describes the data/ properties/attributes and behaviors/methods associated with its instance.
  - It defines data access policy using public, private and protected keywords.

Object life cycle



Access specifiers for data members and methods

- **Access specifiers:**
  - **private** members can be **accessed within class only**
  - **protected** members can be **accessed within class and inside child classes**
  - **default** members can be **accessed within package**
  - **public** members can be **accessed anywhere**
  - **protected** and **default** behaves same within packages but protected can be accessed inside child class

Access specifiers for data members and methods

- Access specifiers/Access Control:

Access Specifier	Same Class	Same Package	Sub-class or child class outside package	Outside class, package and sub-class
public	Yes	Yes	Yes	Yes
protected	Yes	Yes	Yes	No
default	Yes	Yes	No	No
private	Yes	No	No	No

Constructors

- Constructor is special method with same name as class and with or without arguments but no return type.
- **Constructor is used to initialize object**
- If programmer do not write constructor for class, default no argument constructor will be used for object initialization

- **Types of Constructors**
  - **Default constructor (0 or no arguments)**
  - **Parameterized constructor (with arguments)**

Example of class and Constructors


```
package com.cdac.acts;
public class Student {
        private int rollNo;
        private String name;
        private String course;

        public Student() {
        }
        public Student(int rollNo, String name, String course) {
                this.rollNo = rollNo;
                this.name = name;
                this.course = course;
        }
        public void printStudent() {
                System.out.println("\n*****Student Data starts******");
                System.out.println("Student Roll No : "+ rollNo);
                System.out.println("Student Name : "+ name);
                System.out.println("Student course : "+ course);
                System.out.println("*****Student Data ends******");
        }

        public static void main(String[] args) {
                Student s1 = new Student();
                s1.printStudent();
                Student s2 = new Student(123, "Ram", "DAC");
                s2.printStudent();
        }
}
```

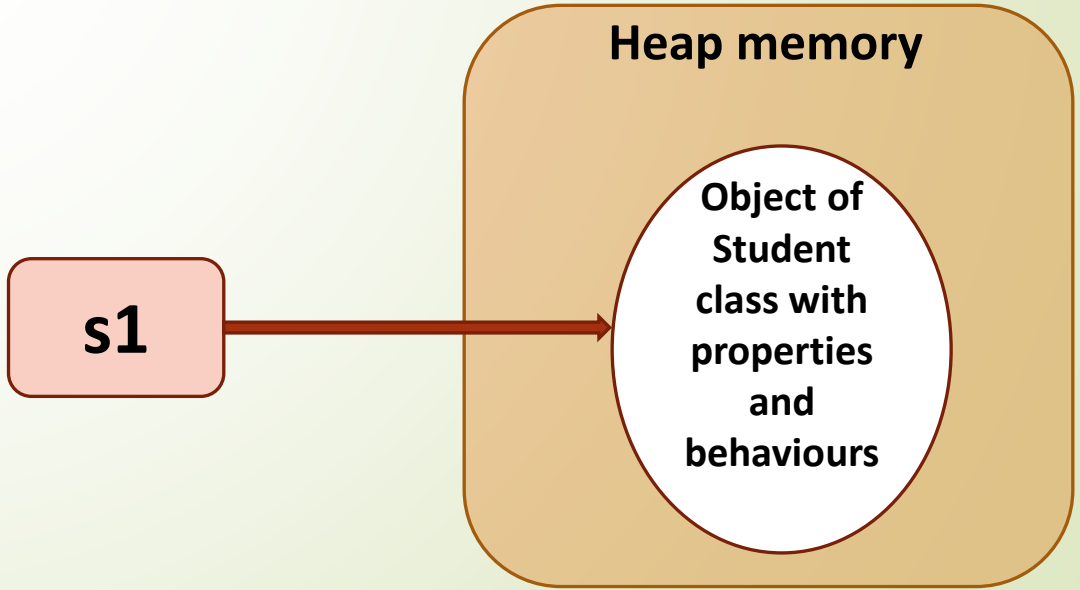
Reference variable

- When we create an object (instance) of class then space is reserved in heap memory.
- We create a reference pointing to object is called **Reference variable**
- Classes, interfaces, arrays, enumerations, and, annotations are reference types in Java.
- Reference variables hold the objects/values of reference types in Java.
- Reference variable can also store **null** value. Default value is **null**

```
Student s1 = new Student(123, "Ram", "DAC");
```

S1 -> Reference variable(located on stack)

Student Object : located on Heap



Reference variable

- Reference can be passed to methods for pass by reference.
- **this** is reference variable which points/refers to invoking objects in instance methods and constructors.
- To be precise **this** is final reference which can not ne changed or modified
- Ex. **this** = null; //invalid statement