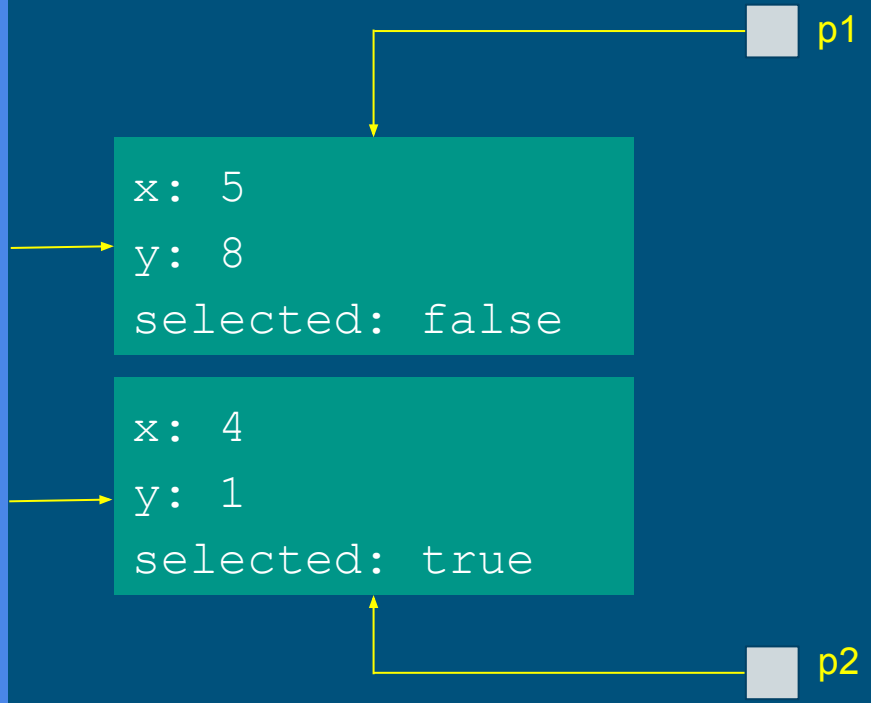
5. Životni ciklus objekta

Sadržaj

- Pojam reference na objekat
- Kreiranje objekta
- Inicijalizacija objekta
- Oslobađanje memorije

Klasa, objekti i reference

```
public class Point {
  private int x, y;
  private boolean selected;
  public void setX(int x) {
    this.x = x;
  }
  public int getX() {
    return this.x;
  }
  ...
}
```



Klase i objekti u programu

- Objekat se kreira korišćenjem operatora `new`
- Prilikom kreiranja objektu se dodeljuje određeni deo memorije
- Deklarisanja reference:

```
Point p1;
```

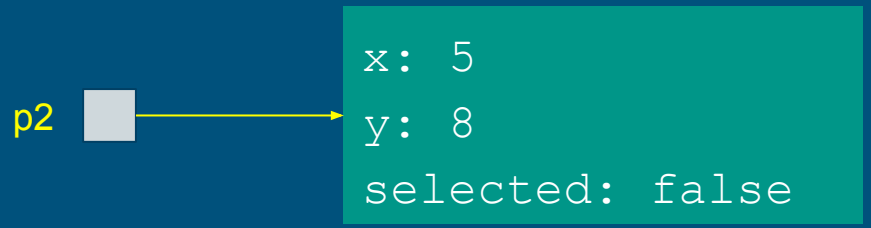
- Deklarisanje reference, kreiranje objekta i inicijalizacija reference:

```
Point p2 = new Point();
```

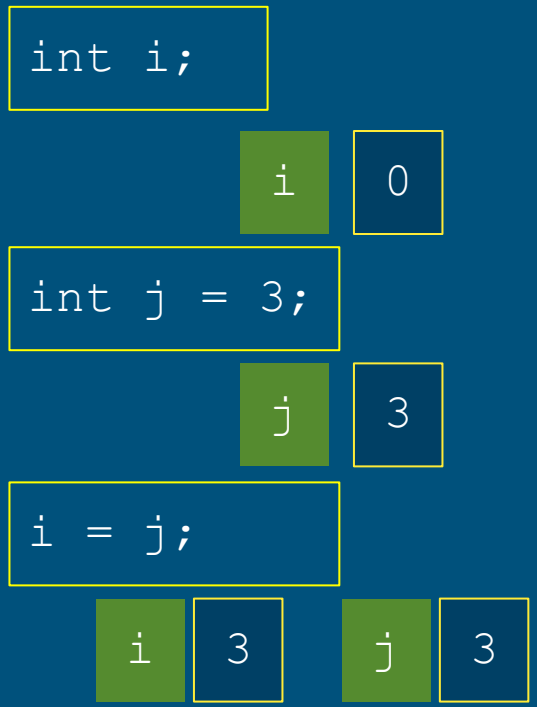
Operator `new`

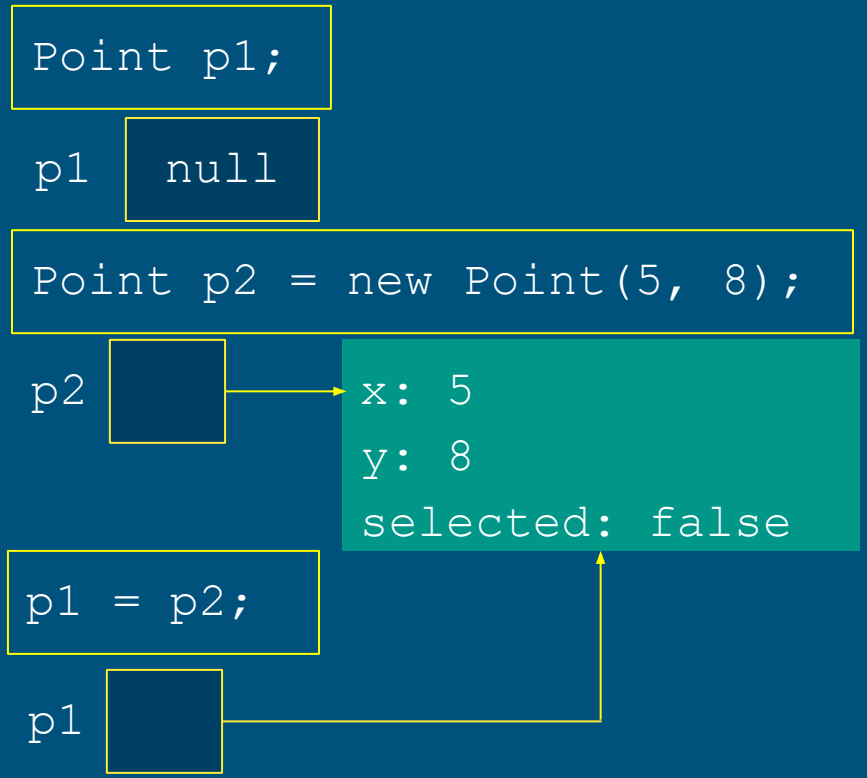
- Alocira i inicijalizuje memoriju za novi objekat
- Poziva konstruktor - specijalnu metodu klase za inicijalizaciju novog objekta
- Vraća referencu na novi objekat

```
Point p2 = new Point(5, 8);
```



Promenljive primitivnog tipa i reference





Referenca `null`

- Može biti dodeljena referenci na objekat
- Referenca na objekat se može porediti sa `null`
- Objekat se može dereferencirati postavljanjem reference na `null`

```
Point p;
…
if (p == null)
  p = new Point(5, 2);
…
p = null;
```

Prosleđivanje parametara

- Parametri primitivnog tipa

```
int x = 5;
refNaObj.nekaMetoda(x);
System.out.println(x);
```

```
public void nekaMetoda(int b) {
  b++;
}
```

- Parametar koji je referenca na objekat

```
Point p1 = new Point(5, 8);
refNaObj.nekaMetoda(p1);
System.out.println(p1.getX());
```

```
public void nekaMetoda(Point p) {
  p.setX(1);
}
```

Inicijalizacija objekta

- Obeležja deklarisana u klasi se inicijalizuju na podrazumevane vrednosti saglasno tipu podatka
  - promenljive primitivnog tipa na podrazumevanu vrednost za dati tip
- Inicijalizacija obeležja na podrazumevane vrednosti najčešće ne zadovoljava potrebe
- Za inicijalizaciju objekta prilikom kreiranja poziva se konstruktor - metoda koja prilikom kreiranja objekta omogućava inicijalizaciju obeležja

Konstruktor

- Metoda koja obezbeđuje inicijalizaciju novokreiranog objekta
- Ima isti naziv kao klasa
- Ne specificira se povratni tip podataka
- Najčešće se deklariše kao `public`
- Ukoliko u klasi nema deklarisan nijedan konstruktor, kompajler generiše podrazumevani konstruktor

Konstruktor u klasi `Point`

```
public class Point {
  private int x, y;
  private boolean selected;

  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
  ...
}
```

Preopterećivanje (engl. *overloading*)

- Deklarisanje više metoda sa istim nazivom i različitim listama parametara

```
public class Point {
  ...
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }
  public Point(int x, int y, boolean selected) {
    this.x = x;
    this.y = y;
    this.selected = selected;
  }
  ...
}
```

Definisanje konstruktora

```
public class Point {
  ...
  public Point(int x, int y) {
    this.x = x;
    this.y = y;
  }

  public Point(int x, int y, boolean selected) {
    this(x, y);
    this.selected = selected;
  }
  ...
}
```

Pozivanje konstruktora

```
public class TestPoint {
  public static void main(String[] args) {
    Point p1 = new Point(5, 8);
    Point p2 = new Point(6, 4, true);

    // podrazumevani konstruktor se ne generise!
    Point p3 = new Point(); // kompajler javlja gresku
  }
}
```

Oslobađanje memorije

- Programu se pri izvršavanju dodele odgovarajući resursi (operativna memorija) koji su ograničeni
- Da bi se tokom izvršenja programa oslobađala memorija, potrebno je da se uklanjaju objekti koji su nepotrebni
- U nekim programskim jezicima postoje metode *destruktori* koje programer mora da obezbedi kako bi se objekti uništavali
- U Javi se o uništavanju objekata brine *garbage collector* koji oslobađa memoriju od svih objekata koji su dereferencirani

Garbage Collector

- Programer ne može da kontroliše oslobađanjem memorije
- Pozivom metode `System.gc()` daje se *hint garbage collector*-u
- Ukoliko objekat koji se uklanja iz memorije zauzima neke resurse (otvoren fajl, na primer) potrebno je osloboditi ih u metodi `finalize()`
- JVM prilikom uklanjanja objekta iz memorije poziva metodu `finalize()`

```
public class Point {
  ...
  public void finalize() {
    System.out.println(“Upravo me uklanja GC!”);
  }
}
```

Rezime

- Objektima se upravalja preko referenci
- Konsturktori obezbeđuju inicijalizaciju objekata prilikom kreiranja
- Klasa može da ima više konstruktora
- Uništavanje objekata i oslobađanje memorije automatski vrši *Garbage Collector*
- Za oslobađanje resursa prilikom uklanjanja objekta iz memorije potrebno je implementirati metodu `finalize()`