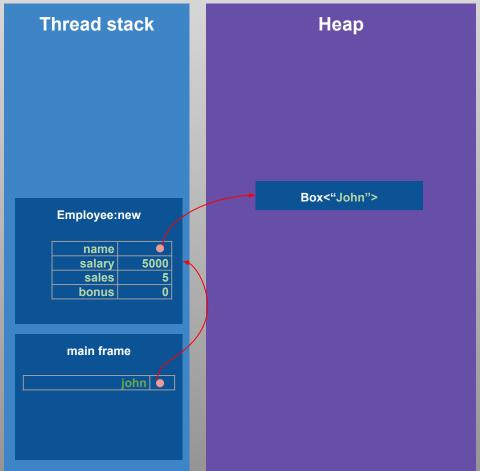
```
struct Employee<'a> {
    name: &'a str,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus_percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
```

# **Thread stack**

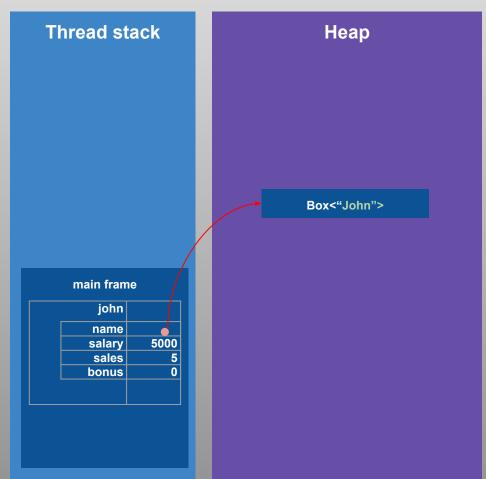
# Heap

main frame

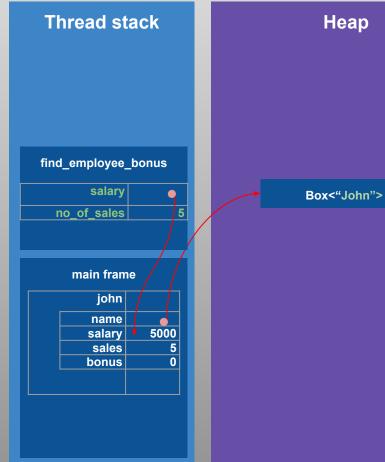
```
struct Employee<'a> {
   name: &'a str,
   sales: i32,
   bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
   let percentage = (salary * BONUS PERCENTAGE) / 100;
   return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
   let bonus_percentage = get_bonus_percentage(salary);
                                                                                        name
   let bonus = bonus percentage * no_of_sales;
                                                                                       salary
   return bonus;
                                                                                        sales
                                                                                       bonus
fn main() {
   let mut john = Employee {
       name: &format!("{}", "John"), // explicitly making the value dynamic
   john.bonus = find_employee_bonus(&john.salary, john.sales);
```



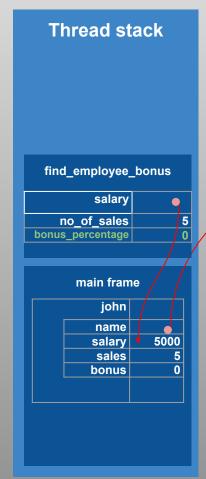
```
struct Employee<'a> {
    name: &'a str,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
```



```
struct Employee<'a> {
    name: &'a str,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus_percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
    john.bonus = find_employee_bonus(&john.salary, john.sales);
```

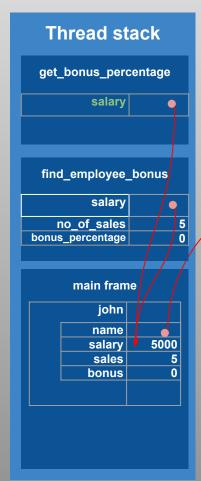


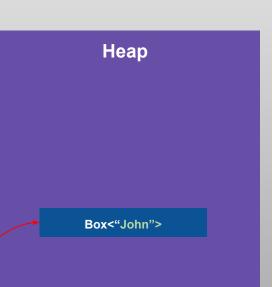
```
struct Employee<'a> {
    name: &'a str,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus_percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
    john.bonus = find_employee_bonus(&john.salary, john.sales);
```



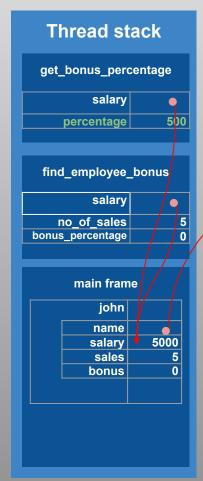


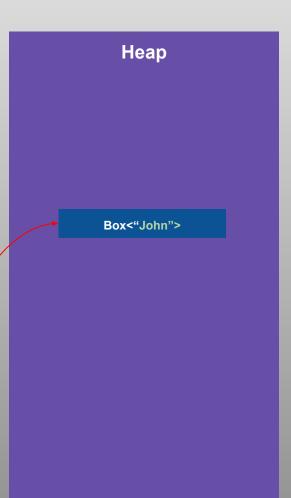
```
struct Employee<'a> {
   name: &'a str,
   salary: i32,
   sales: i32,
   bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
   let percentage = (salary * BONUS PERCENTAGE) / 100;
   return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
   let bonus_percentage = get_bonus_percentage(salary);
   let bonus = bonus_percentage * no_of_sales;
   return bonus;
fn main() {
   let mut john = Employee {
   john.bonus = find_employee_bonus(&john.salary, john.sales);
```



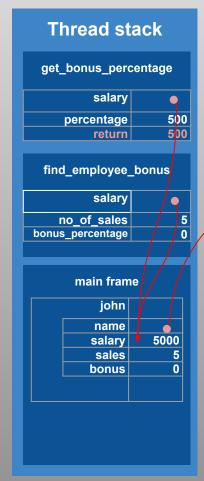


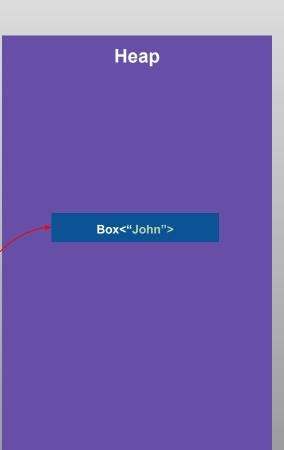
```
struct Employee<'a> {
   name: &'a str,
   salary: i32,
   sales: i32,
   bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
   let percentage = (salary * BONUS PERCENTAGE) / 100;
   return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
   let bonus_percentage = get_bonus_percentage(salary);
   let bonus = bonus_percentage * no_of_sales;
   return bonus;
fn main() {
   let mut john = Employee {
   john.bonus = find_employee_bonus(&john.salary, john.sales);
```



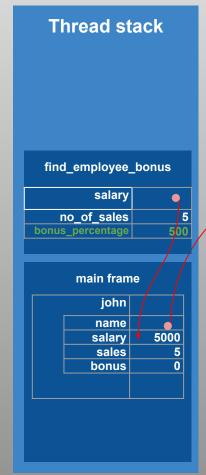


```
struct Employee<'a> {
    name: &'a str,
    salary: i32,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus_percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
        bonus: 0,
    john.bonus = find_employee_bonus(&john.salary, john.sales);
```



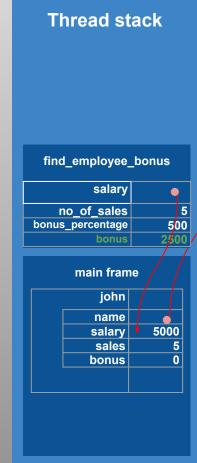


```
struct Employee<'a> {
    name: &'a str,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus_percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
    john.bonus = find_employee_bonus(&john.salary, john.sales);
```



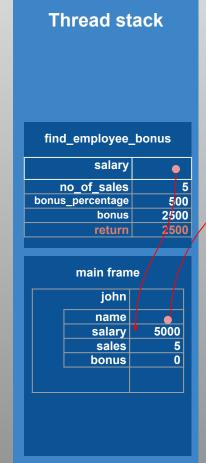


```
struct Employee<'a> {
    name: &'a str,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus percentage * no of sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
    john.bonus = find_employee_bonus(&john.salary, john.sales);
```



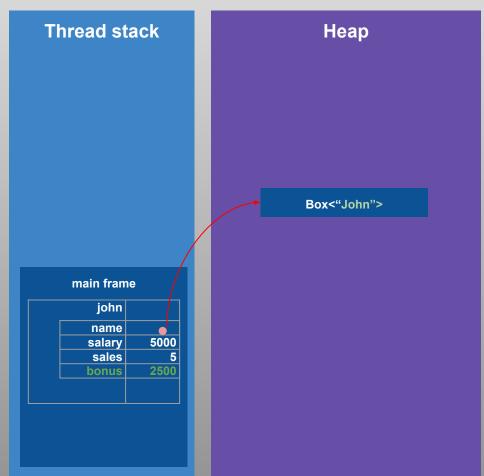


```
struct Employee<'a> {
    name: &'a str,
    salary: i32,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus percentage * no of sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
    john.bonus = find_employee_bonus(&john.salary, john.sales);
```





```
struct Employee<'a> {
    name: &'a str,
    sales: i32,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
```



```
struct Employee<'a> {
    name: &'a str,
    bonus: i32,
const BONUS_PERCENTAGE: i32 = 10;
fn get bonus percentage(salary: &i32) → i32 {
    let percentage = (salary * BONUS PERCENTAGE) / 100;
    return percentage;
fn find employee bonus(salary: &i32, no of sales: i32) → i32 {
    let bonus_percentage = get_bonus_percentage(salary);
    let bonus = bonus percentage * no_of_sales;
    return bonus;
fn main() {
    let mut john = Employee {
        name: &format!("{}", "John"), // explicitly making the value dynamic
```

#### Thread stack

# Heap